



Team AmigoBot™ Operations Manual

*with
AmigoSH Robot Control Software,
SonARNL Localization and Navigation Servers,
ARIA Robotics Development Software and MobileEyes™ LAN Client.*

© 2006 MobileRobots Inc. All rights reserved.

This document, as are the software described in it, is provided under license and may only be used or copied in accordance with the terms of the respective license.

Information in this document is subject to change without notice and should not be construed as a commitment by MobileRobots Inc.

The software on disk, CD-ROM and firmware which accompany the robot and are available for network download by MobileRobots customers are solely owned and copyrighted or licensed for use and distribution by MobileRobots Inc.

Developers and users are authorized by revocable license to develop and operate custom software for personal research and educational use only. Duplication, distribution, reverse-engineering or commercial application of MobileRobots software and hardware without license or the express written consent of MobileRobots Inc. is explicitly forbidden.

PeopleBot™, AmigoBot™, PowerBot™, PatrolBot™, ARCSinside™, SetNetGo™, MobilePlanner™, MobileSim™ and MobileEyes™ are trademarks of MobileRobots Inc. Other names and logos for companies and products mentioned or featured in this document are often registered trademarks or trademarks of their respective companies. Mention of any third-party hardware or software constitutes neither an endorsement nor a recommendation by MobileRobots Inc.

Important Safety Instructions

- ✓ Read the installation and operations instructions before using the equipment.
- ✓ Avoid using power extension cords.
- ✓ To prevent fire or shock hazard, do not expose the equipment to rain or moisture.
- ✓ Refrain from opening the unit or any of its accessories.
- ✓ Keep wheels away from long hair or fur.
- ✓ Never access the interior of the robot with charger attached or batteries inserted.

Inappropriate Operation

Inappropriate operation voids your warranty! Inappropriate operation includes, but is not limited to:

- ✓ Dropping the robot, running it off a ledge, or otherwise operating it in an irresponsible manner
- ✓ Overloading the robot above its payload capacity
- ✓ Getting the robot wet
- ✓ Continuing to run the robot after hair, yarn, string, or any other items have become wound around the robot's axles or wheels
- ✓ Opening the robot with charger attached and/or batteries inserted
- ✓ All other forms of inappropriate operation or care

**Use MOBILEROBOTS authorized parts *ONLY*;
warranty void otherwise.**

Table of Contents

CHAPTER 1 INTRODUCTION	7
ROBOT PACKAGES	7
<i>Basic Components (all shipments).....</i>	<i>7</i>
<i>Optional Components and Attachments (partial list).....</i>	<i>7</i>
<i>User-Supplied Components / System Requirements.....</i>	<i>7</i>
ADDITIONAL RESOURCES	7
<i>Support Website</i>	<i>8</i>
<i>Newsgroups</i>	<i>8</i>
<i>Support.....</i>	<i>8</i>
CHAPTER 2 WHAT IS AMIGOBOT?	9
PIONEER REFERENCE PLATFORM	9
PIONEER FAMILY OF ROBOT CONTROLLERS AND OPERATIONS SOFTWARE	9
PORTS AND POWER	10
CLIENT SOFTWARE	10
<i>ARIA and ARNetworking.....</i>	<i>10</i>
<i>Mapping, Localization, and Navigation</i>	<i>11</i>
MODES OF OPERATION	11
<i>Server Mode.....</i>	<i>11</i>
<i>Maintenance and Standalone Modes.....</i>	<i>11</i>
THE PIONEER LEGACY.....	12
<i>Pioneer AT</i>	<i>12</i>
<i>Pioneer 2™ and PeopleBot™</i>	<i>12</i>
<i>AmigoBot.....</i>	<i>13</i>
<i>Pioneer 3 and Recent Pioneer 2-DX8, -AT8 and Plus Mobile Robots</i>	<i>13</i>
<i>Pioneer 3-SH Robots</i>	<i>13</i>
CHAPTER 3 SPECIFICATIONS & CONTROLS	14
PHYSICAL CHARACTERISTICS AND COMPONENTS	14
POWER AND CHARGING	15
USER INDICATORS AND CONTROLS	15
<i>Serial Communications & Expansion I/O.....</i>	<i>16</i>
SONAR.....	16
<i>Sonar Rate and Sequence</i>	<i>16</i>
<i>Sonar Sensitivity</i>	<i>17</i>
MOTORS, WHEELS AND ENCODERS.....	17
SAFETY AMIGO SH WATCHDOGS	17
CHAPTER 4 QUICK START AND SELF-TEST	18
PREPARATIVE ASSEMBLY.....	18
SELF TEST.....	18
CLIENT DEMONSTRATIONS	18
CLIENT-SERVER COMMUNICATIONS	19
TETHERED WITH ARIA DEMO	19
<i>Demo Startup Options.....</i>	<i>19</i>
<i>A Successful Connection.....</i>	<i>20</i>
<i>Operating the ARIA Demo</i>	<i>20</i>
<i>Disconnecting.....</i>	<i>21</i>
NETWORKING WITH MOBILEEYES	21
<i>Start serverDemo</i>	<i>21</i>
<i>Start MobileEyes and Connect with serverDemo</i>	<i>21</i>
<i>Operating MobileEyes</i>	<i>22</i>
TROUBLESHOOTING	22
CHAPTER 5 AMIGOSH.....	23
CLIENT-SERVER COMMUNICATION PACKET PROTOCOLS	23
<i>Packet Checksum</i>	<i>24</i>
<i>Packet Errors</i>	<i>24</i>
THE CLIENT-SERVER CONNECTION	25
<i>Autoconfiguration (SYNC2)</i>	<i>25</i>

Opening the Servers—OPEN	25
Server Information Packets	25
Keeping the Beat—PULSE	26
Closing the Connection—CLOSE	27
CLIENT COMMANDS	27
Client Motion Commands	29
ROBOTS IN MOTION	29
PID Controls	30
Position Integration	31
DriftFactor, RevCount and TicksMM	31
SONAR	31
Enable/Disabling Sonar	31
Polling Sequence	32
Polling Rate	32
STALLS AND EMERGENCIES	32
ACCESSORY COMMANDS AND PACKETS	32
Packet Processing	32
CONFIGpac and CONFIG Command	32
SERIAL	34
HOST-to-AUX Serial Transfers	34
ENCODERS	34
BUZZER SOUNDS	34
TCM2	35
Heading Correction Gyro	35
CHAPTER 6 UPDATING & RECONFIGURING AMIGOSH	36
WHERE TO GET AMIGOSH SOFTWARE	36
AMIGOSH MAINTENANCE MODE	36
Enabling Maintenance Mode on the Controller	36
AMIGOSHCF	36
STARTING AMIGOSHCF	37
CONFIGURING AMIGOSH PARAMETERS	37
Interactive Commands	38
Changing Parameters	38
SAVE YOUR WORK	38
PID PARAMETERS	39
DRIFTFACTOR, TICKSMM AND REVCOUNT	40
STALLVAL AND STALLCOUNT	40
CHAPTER 7 CALIBRATION & MAINTENANCE	41
CALIBRATING YOUR ROBOT	41
DRIVE LUBRICATION	41
AMIGOBOT BATTERIES	41
Charging the Battery	42
Alternative Battery Chargers	42
GETTING INSIDE	42
FACTORY REPAIRS	42
APPENDIX A	44
CONTROLLER PORTS & CONNECTIONS	44
AMIGOSH CONTROLLER	44
Battery and Motors Connector	44
Serial Ports	45
Encoders	45
Sonar	45
Buttons and Indicators	46
Heading Correction Gyro	46
Expansion I/O	46
APPENDIX B	47
SPECIFICATIONS	47
WARRANTY & LIABILITIES	49

Chapter 1 Introduction

Congratulations on your purchase and welcome to the rapidly growing community of developers and enthusiasts of MOBILEROBOTS intelligent mobile platforms.

This *Team AmigoBot™ Operations Manual* provides both the general and technical details you need to operate your new AmigoBot robots and to begin developing your own robotics hardware and software.

For operation of previous versions of AmigoBot, which use the Hitachi H8S-based controller and support software, please contact sales@activmedia.com or access our support website: <http://robots.amigobot.com> for related documentation.



Figure 1. AmigoBots prefer to swarm into teams.

ROBOT PACKAGES

Our experienced manufacturing staff put your mobile robot and accessories through a “burn in” period and carefully tested them before shipping the products to you. In addition to the companion resources listed above, we warrant your MOBILEROBOTS platform and our manufactured accessories against mechanical, electronic, and labor defects for one year. Third-party accessories are warranted by their manufacturers, typically for 90 days.

Even though we’ve made every effort to make your AmigoBot package complete, please check the components carefully after you unpack them from the shipping crate.

Basic Components (all shipments)

- ✓ One fully assembled Team AmigoBot robot with charged battery
- ✓ CD-ROM containing licensed copies of MOBILEROBOTS software and documentation
- ✓ Set of manuals
- ✓ Registration and Account Sheet

Optional Components and Attachments (partial list)

- ✓ Battery charger (some contain power receptacle and 220VAC adapters)
- ✓ Radio Ethernet
- ✓ Supplementary and replacement batteries
- ✓ Pan-Tilt-Zoom Surveillance Cameras
- ✓ Heading-correction gyro
- ✓ Compass
- ✓ Serial cables for external connections
- ✓ Many more...

User-Supplied Components / System Requirements

- ✓ Client PC: 586-class or later PC with Microsoft Windows© or Linux OS
- ✓ One RS-232 compatible serial port or Ethernet (requires Ethernet-to-serial accessory)
- ✓ Four megabytes of available hard-disk storage

ADDITIONAL RESOURCES

Every MOBILEROBOTS customer gets three additional and valuable resources:

- ✓ A private account on our support Internet website for downloading software, updates, and manuals
- ✓ Access to private newsgroups
- ✓ Direct access to the MOBILEROBOTS technical support team

Support Website

We maintain a 24-hour, seven-day per week World Wide Web server where customers may obtain software and support materials:

<http://robots.MobileRobots.com/amigobot>

Some areas of the website are restricted to licensed customers. To gain access, enter the username and password written on the *Registration & Account Sheet* that accompanied your robot.

Newsgroups

We maintain several email-based newsgroups through which robot owners share ideas, software, and questions about the robot. Visit the support <http://robots.MobileRobots.com> website for more details. To sign up for amigobot-users, for example, send an e-mail message to the `-requests` automated newsgroup server:

To: **amigobot-users-requests@MobileRobots.com**
From: <your return e-mail address goes here>
Subject: <choose one command:>
 help (returns instructions)
 lists (returns list of newsgroups)
 subscribe
 unsubscribe

Our e-mail list server will respond automatically. After you subscribe, e-mail your comments, suggestion, and questions intended for the worldwide community of AmigoBot users:¹

To: **amigobot-users@MobileRobots.com**
From: <your return e-mail address goes here>
Subject: **<something of interest to pioneer users>**

Access to the `amigobot-users` e-mail newslst is limited to subscribers, so your address is safe from spam. However, the list currently is unmoderated, so please confine your comments and inquiries to issues concerning the operation and programming of MOBILEROBOTS platforms.

Support

Have a problem? Can't find the answer in this or any of the accompanying manuals? Or do you know a way that we might improve our robots? Share your thoughts and questions with us from the online form at the support website:

<http://robots.MobileRobots.com/techsupport>

or by email:

support@MobileRobots.com

Please include your robot's **serial number** (look for it beside the Main Power switch)—we often need to understand your robot's configuration to best answer your question.

Tell us your robot's SERIAL NUMBER.

Your message goes directly to the MOBILEROBOTS technical support team. There a staff member will help you or point you to a place where you can find help. Because this is a support option, not a general-interest newsgroup like `amigobot-users`, we reserve the option to reply only to questions about problems with your robot or software.

¹ Note: Leave out the `-requests` part of the email address when sending messages to the newsgroup.

Chapter 2 What Is AmigoBot?

Team AmigoBot™ is the smallest member of the Pioneer family of mobile robots, including the Pioneer 1 and Pioneer AT, Pioneer 2-DX, -DXe, -DXf, -CE, -AT, -DX8/Dx8 *Plus* and -AT8/AT8 *Plus*, and the newest Pioneer 3-DX and -AT mobile robots. These small, research and development platforms share a common architecture and foundation software with all other MOBILEROBOTS platforms, including PeopleBot™ V1, Performance PeopleBot™, PatrolBot™ and PowerBot™.

PIONEER REFERENCE PLATFORM

MOBILEROBOTS platforms set the standards for intelligent mobile robots by containing all of the basic components for sensing and navigation in a real-world environment. They have become reference platforms in a wide variety of research projects, including several US Defense Advanced Research Projects Agency (DARPA) funded studies.

Every MOBILEROBOTS platform comes complete with a sturdy aluminum body, balanced drive system (two-wheel differential with casters or four-wheel skid-steer), reversible DC motors, motor-control and drive electronics, high-resolution motion encoders, and battery power, all managed by an onboard controller and mobile-robot server software.

Besides the open-systems robot-control server software onboard the robot controller, every MOBILEROBOTS platform also comes with a host of advanced robot-control client software applications and applications-development environments. Software development includes our own foundation Advanced Robotics Interface for Applications (ARIA) and ArNetworking, released under the GNU Public License, and complete with fully documented C++, Java, and Python libraries and source code. Several third-party robotics applications development environments also have emerged from the research community, including Saphira from SRI International, Ayllu from Brandeis University, Pyro from Bryn Mawr and Swarthmore Colleges, Player/Stage from the University of Southern California, and Carmen from Carnegie-Mellon University.

PIONEER FAMILY OF ROBOT CONTROLLERS AND OPERATIONS SOFTWARE

First introduced in 1995, the original Pioneer 1 mobile robot contained a controller based on the Motorola 68HC11 microprocessor and powered by Pioneer Server Operating System (PSOS) software. The next generation of Pioneer 2 and PeopleBot robots used a Siemens C166-based controller with Pioneer 2 Operating System (P2OS) software. AmigoBot introduced an Hitachi H8S-based controller with AmigOS in 2000. From 2002 until Fall of 2004, Pioneer 3, Performance PeopleBot, and PowerBot robots also had an Hitachi H8S-based controller with ActivMedia Robotics Operating System (AROS) software.

Now, all MOBILEROBOTS platforms use revolutionary high-performance controllers with advanced embedded robot control software based on the new-generation 32-bit Renesas SH2-7144 RISC microprocessor, including the P3-SH controller with ARCOS, μ ARCS*Inside* the PatrolBot and our industrial Core, and AmigoSH for AmigoBot.

But you might not even notice the differences. Because we have taken great care to ensure backward compatibility across ActivMedia/MOBILEROBOTS entire history of robots, client software written to operate an ancient PSOS-based Pioneer AT will work with a brand new Team AmigoBot with little or no modification. Client-server communication over a serial communication link remain identical as do support for all robotics commands.

See Chapter 5, *AmigoSH*, for details.



Figure 2. MOBILEROBOTS

PORTS AND POWER

Your Team AmigoBot has a variety of expansion power and I/O ports for attachment and close integration of a client PC, sensors, and a variety of accessories—all accessible through a common application interface to the robot's server software, AmigoSH. Features include:

- ✓ 44.2368 MHz Renesas SH2 32-bit RISC microprocessor with 32K RAM and 128K FLASH
- ✓ 2 RS-232 serial and 2 SCI ports configurable from 9.6 to 115.2 kilobaud
- ✓ 1 Array of 8 sonar
- ✓ Heading correction gyro port
- ✓ User Control Panel with RESET and MOTORS pushbutton controls; Power, System and User LEDs, and a piezo buzzer
- ✓ 4-line X 20-character LCD support
- ✓ Expansion connector with built-in 8-bit r/w bus, analog and digital I/O and more...

CLIENT SOFTWARE

All MOBILEROBOTS platforms, including AmigoBot, operate as the server in a client-server environment: Their controllers handle the low-level details of mobile robotics, including maintaining the platform's drive speed and heading over uneven terrain, acquiring sensor readings, such as from the sonar, and managing attached accessories. To complete the client-server architecture, AmigoBot requires a PC connection: software running on a computer connected with the robot's controller via the HOST serial link and which provides the high-level, intelligent robot controls, including obstacle avoidance, path planning, features recognition, sonar localization, navigation, and so on.

An important benefit of MOBILEROBOTS' client-server architecture is that different robot servers can be run using the same high-level client. Several clients also may share responsibility for controlling a single mobile server, which permits experimentation in distributed communication, planning and control.

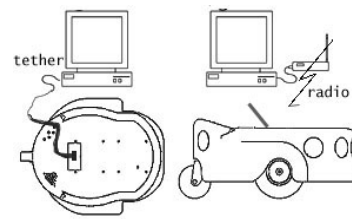


Figure 3. Team AmigoBot requires a PC to run client software

ARIA and ARNetworking

Advanced Robotics Interface for Applications (ARIA) software, including ARNetworking, comes with every MOBILEROBOTS platform. ARIA is a C++-based open-source development environment that provides a robust client-side interface to a variety of intelligent robotics systems, including your robot's controller and accessory systems.

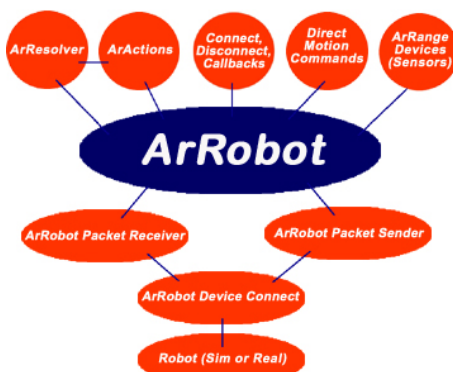


Figure 4. ARIA

ARNetworking provides the critical layer for TCP/IP-based communications with your MOBILEROBOTS platform over the network. ARIA is the ideal platform for integration of your own robot-control software, since it neatly handles the lowest-level details of client-server interactions, including serial communications, command and server-information packet processing, cycle timing, and multithreading, as well as support of a variety of accessories and controls, such as a scanning laser-range finder, motion gyros, sonar, and many others.

What's more, ARIA and ARNetworking come with source code so that you may examine the software and modify it for your own sensors and applications.

Mapping, Localization, and Navigation

MOBILEROBOTS also provides a comprehensive suite of client tools and applications by which you create, edit, and use maps and floor plans for advanced robotics applications, including localization and navigation. These include the MobileEyes, Mapper3Basic and SonARNL software that come with your Team AmigoBot. Please see the *SonARNL* documentation for details.

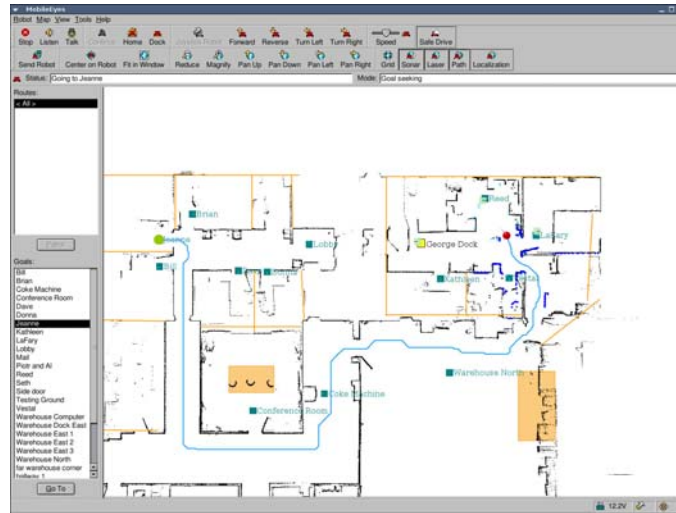


Figure 5. Use MobileEyes with SonARNL for advanced operation of Team AmigoBot over the network.

MODES OF OPERATION

You may operate your AmigoBot in one of three modes:

- ✓ Server
- ✓ Maintenance
- ✓ Standalone

Server Mode

Your AmigoBot's new Renesas SH2-7144 microprocessor-based controller comes with 128K of re-programmable FLASH and 32K dynamic RAM memory. However, we don't recommend that you start learning SH2 programming. Rather, the robot comes to you installed with the latest AmigoSH servers so you may concentrate on your clients.

In conjunction with client software like ARIA running on an onboard or other user-supplied computer, AmigoSH (pronounced "ah-me-gosh") lets you take advantage of modern client-server and robot-control technologies to perform advanced robot tasks. Most users run their MOBILEROBOTS platform in server mode, because it gives them quick, easy access to its robotics functionality while working with high-level software on a familiar host computer.

Maintenance and Standalone Modes

For experiments in controller-level operation of your robot's functions, you may reprogram its onboard FLASH for direct and standalone operation. We supply the means to download and debug (AMIGOSTub embedded GDB interface), but not the controller's programming software, for you to work in standalone mode.

The utilities we provide for you to reprogram the AmigoBot's controller FLASH also may be used to update and upgrade your robot's AmigoSH servers. In a special maintenance mode, you also adjust

your robot's operating parameters that AmigoSH uses as default values on startup, during a client connection, and on reset. See Chapter 6, *Updating & Reconfiguring AmigoSH*, for much more detail.

We typically provide the maintenance utilities and AmigoSH upgrades free for download from our website, so be sure to sign up for the `amigobot-users` email newlist. That's where we notify our customers of the upgrades, as well as where we provide access to MOBILEROBOTS users worldwide.

THE PIONEER LEGACY

Commercially introduced in the summer of 1995, Pioneer 1 was the original platform. Intended mostly for indoor use on hard, flat surfaces, the robot had solid rubber tires and a two-wheel differential, reversible drive system with a rear caster for balance. It came with a single-board 68HC11-based robot controller and the Pioneer Server Operating System (PSOS) software. The Pioneer 1 also came standard with seven sonar range finders (two side-facing and five forward-facing) and integrated wheel encoders. Its low-cost and high-performance caused an explosion in the number of researchers and developers who now have access to a real, intelligent mobile robotic platform.



Figure 6. Pioneer 1 was first introduced to the world at IJCAI '95 in Montreal.

Software-wise, the Pioneer 1 initially served as a platform for SRI International's AI/fuzzy logic-based Saphira robotics applications development. But it wasn't long before Pioneer's open architecture became the popular platform for the development of a variety of alternative robotics software environments.

Pioneer AT

Functionally and programmatically identical to the Pioneer 1, the four-wheel drive, skid-steer Pioneer AT was introduced in the summer of 1997 for operation in uneven indoor and outdoor environments, including loose, rough terrain.

Except for the drive system, there were no operational differences between the Pioneer AT and the Pioneer 1: The integrated sonar arrays and controllers were the same; they shared accessories; and applications developed for the Pioneer 1 worked with little or no porting on the AT.

Pioneer 2™ and PeopleBot™

The next generation of Pioneer, including the Pioneer 2-DX, -CE, and -AT, introduced in fall of 1998 through summer of 1999, improved upon the Pioneer 1 legacy while retaining its many important advantages.² Indeed, in most respects, particularly with applications software, Pioneer 2 worked identically to Pioneer 1 models, but offered many more expansion options, including a client PC onboard the robot.

The ActivMedia Robotics Pioneer 2 models -DX, -DE, -DXe, -DXf, and -AT, and the V1 and Performance PeopleBot robots used a 20 MHz Siemens 88C166-based controller, with independent motor-power and sonar controller boards for a versatile operating environment. Sporting a more holonomic body, larger wheels and stronger motors for better indoor performance, Pioneer 2-DX, -DXe, -DXf, and -CE models, like Pioneer 1, were two-wheel, differential-drive mobile robots.

The four-wheel drive Pioneer 2-AT had four independent motors and drivers. Unlike its Pioneer AT predecessor, the Pioneer 2-AT came with a stall-detection system and inflatable pneumatic tires with metal wheels for much more robust operation in rough terrain, as well as the ability to carry nearly 30 kilograms (66 lbs) of payload and climb a 60-percent grade.

Other Pioneer 2-like robots include the Performance PeopleBot robots, which were introduced in 2000. They are architecturally Pioneer 2 robots, but with stronger motors and integrated human-interaction features, including a pedestal extension, integrated voice and sound synthesis and

² Price/performance ratio included! The much more capable and expandable Pioneer 2 was introduced four years later for just a few hundred dollars (US) more than the original Pioneer 1.

recognition—ideal for human-interaction studies as well as for commercial and consumer mobile-robotics applications.

AmigoBot

Although sporting a 16-bit Hitachi H8S-based controller and highly durable polycarbonate body, the AmigoBot in many ways was an updated Pioneer 1: Released in Summer of 2000, it reintroduced the concept of a small, lightweight, self-contained platform ideal for training and education in intelligent mobile robot controls.

Like the original, your new Team AmigoBot, first released in Spring of 2005, has a single controller board for client-server operation with motor drives, sonar, and other device supports built in. The principle difference in design from the original model is that the new AmigoBot sports the advanced SH2-based controller.

Pioneer 3 and Recent Pioneer 2-DX8, -AT8 and Plus Mobile Robots

Two new models of Pioneer 2 appeared in the summer of 2002, two more at the beginning of 2003, and the Pioneer 3 debuted in the summer of 2003. All used a special “P2H8” controller based on the Hitachi H8S microprocessor, with new control systems software (AROS) and I/O expansion capabilities. The Pioneer 3 and 2-*Plus* robots also had new, more powerful motor/power systems for better navigational control and payload.³

Pioneer 3-SH Robots

Like your new Team AmigoBot, the latest Pioneer, Performance PeopleBot, and PowerBot robots—all introduced in summer of 2004—are identical to their predecessors except for their revolutionary new Renesas SH2-based controller. Software-wise, these new robots are fully compatible with all other MOBILEROBOTS platforms, including Pioneer 1. Their new Advanced Robot Control & Operation Software (ARCOS) provides unprecedented performance and expansion, yet can interface and run client programs originally developed for Pioneer 1, 2, as well as 3 platforms.

To the relief of those who have invested years in developing software for MOBILEROBOTS platforms, your new Team AmigoBot truly does combine the best of the new mobile robot technologies with MOBILEROBOTS’ tried-and-true robot architecture.



Figure 7. PowerBot™ carries over 100 kg of payload.

³ The interim Pioneer 2-DXf had the same, more-powerful motors as the DX8s and AT8 *Plus*.

Chapter 3 Specifications & Controls

AmigoBot is an intelligent mobile robot. Combined with state-of-the-art robotics software like ARIA, ARNetworking, SONARNL and MobileEyes running on your PC, AmigoBot can actually determine where it is (localization) and find its way safely from one place to the next (navigation); with your help (teleoperation) and without your help (self-guided navigation), all the while going around obstacles, through doors, over bumps, under tables...

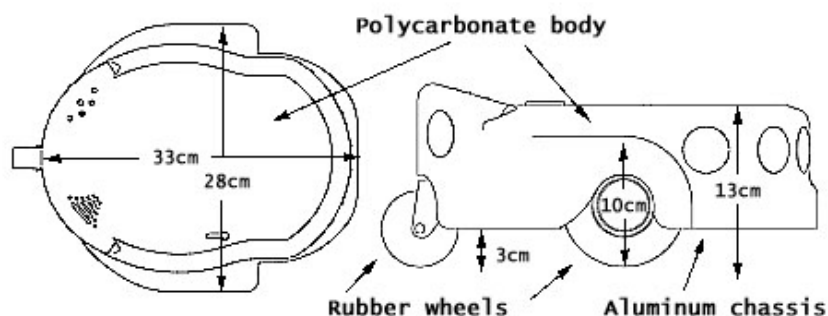


Figure 8. AmigoBot features

PHYSICAL CHARACTERISTICS AND COMPONENTS

AmigoBot is intended for indoor use in wheelchair-accessible places, including homes, schools, nursing homes, hospitals, offices, and research labs. The small (28 x 33 x 13 centimeters), lightweight (3.6 kilograms with batteries), and highly maneuverable (meters-per-second translation; 300 degrees-per-second rotation; turns in place) AmigoBot robot has a high impact-resistant polycarbonate body with a solid, but lightweight aluminum chassis. Each of its two solid 10.2 centimeter-diameter rubber tires are driven by a reversible 12 VDC motor. The drive system includes a passive rear caster for balance.

The standard AmigoBot comes with eight range-finding sonar: one on each side, four forward facing and two in the rear for 360-degree sensing coverage.

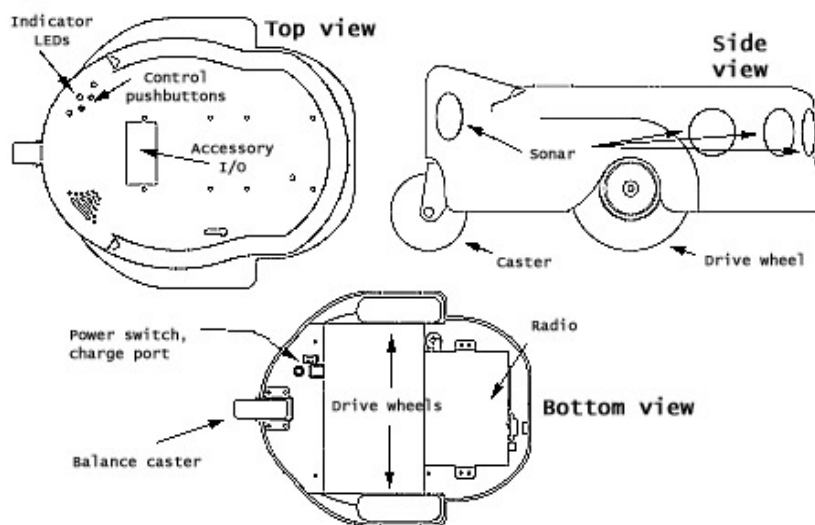


Figure 9. AmigoBot's sensors, controls, indicators and ports

Attached to each drive axle is a high-resolution optical quadrature shaft encoder that provides 123 ticks-per-millimeter of wheel rotation for precise position and speed sensing and advanced dead-reckoning.

The AmigoBot drive and sensor systems are powered and processed from a single controller, driven by a high-performance, I/O-rich Renesas SH2 microprocessor. The resident operating system (AmigoSH) has 16K RAM and 128K FLASH on-chip memory to work with.

POWER AND CHARGING

A single slide-switch on the bottom of AmigoBot near the caster controls power to the entire robot and all of its integrated accessories. A red LED on the top towards the rear of the robot is lit when the AmigoBot has power.

The standard AmigoBot comes with a pair of 12 VDC sealed lead/acid batteries for a total of 4.6 ampere-hours (55.2 watt-hours)—more than ample power for its drives, electronics, and accessories. Under typical operation with continuous motor activity, the battery provides over 3 hours of service.

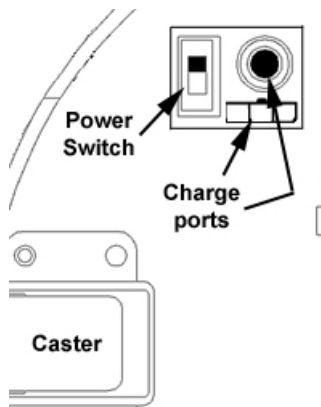


Figure 10. AmigoBot's power switch and charge ports

Use the recommended accessory 12 VDC charger or equivalent and plug into the round charge port to recharge the lead-acid batteries and to power your AmigoBot.⁴ All the recommended chargers have sufficient power to run your AmigoBot's controller as well as charge the internal batteries, so you need not turn off power.

You should maintain AmigoBot's batteries in a charged state above 11 VDC. We recommend recharging the battery when it falls below 11.7 VDC (`LowBattery FLASH` parameter), even though the robot may continue to operate below 11 VDC. The controller will sound a warning when the battery voltage falls below that programmed level and will automatically shut down the motors and any active client-server connection when the battery voltage falls below a `ShutdownVolts FLASH` parameter's value so to avoid damage.

See Chapter 5, *Updating and Reconfiguring AmigoSH* for additional details.

Disengage the motors when recharging AmigoBot, but you may continue operating the robot.

Typical recharge time depends on the charger and the discharge state of the battery. The AmigoBot's standard charger (800 milliampere two-state fast charger) recharges the lead-acid battery in 2-4 hours. You may continue to operate AmigoBot while charging its batteries, although that will lengthen the recharge time.

USER INDICATORS AND CONTROLS

On top, near the back of AmigoBot next to the red power LED, are two pushbutton switches and two additional LEDs. The red pushbutton switch is the `Reset` button. Press it at any time to reset the AmigoBot controller to its power start-up state—motors disengaged and not connected with a client.

The black `Motors/Test` pushbutton's function depends on the operating state of the robot. When not connected with a client, press the button once to enable and a second time to begin AmigoBot's self-tests, for instance.

See Chapter 3, *Quick Start and Self Tests*, for details.

⁴ The three-position charge port is for future battery styles and requisite charger.

Specifications & Controls

When connected with a client, the black `Motors/Test` button manually enables and disables the motors, which also can be performed programmatically with the AmigoSH command #4, as described in detail in Chapter 4.

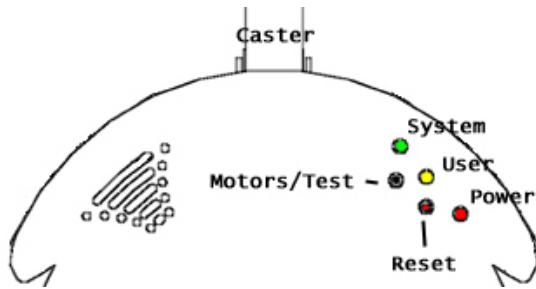


Figure 11. AmigoBot's control buttons and indicators

When pressed in combination, the `Motors/Test` and `Reset` buttons enable Maintenance Mode on the controller. The green `System` and amber `User` LEDs indicate AmigoSH activities depending on the current mode of operation. For example on start up, the green LED flashes slowly and rhythmically, while the amber LED is OFF. When connected with a client, the amber `User` lamp flashes rapidly indicating client-server serial communications activity. And when connected, the green `System` LED flashes rapidly when the motors are enabled, and slowly when the motors are disabled.

A built-in piezo buzzer, audible through the holes on the right top side of AmigoBot across from the indicators and controls, provides audible clues to the robot's state, such as upon successful startup of the controller and a client connection. An AmigoSH client command lets you program the buzzer, too, to play your own MIDI sounds.

Serial Communications & Expansion I/O

Every AmigoBot comes with a 9-pin DSUB serial connector and power plug that emerge from the bottom of the robot. These normally are for connection to the Ethernet-to-serial accessory for wireless client-server connections.

Otherwise, for AmigoSH FLASH maintenance and for tethered client-server communication with the controller, you also access AmigoBot's `Host` (RS-232) serial port by lifting out and removing the black rectangular plastic cover near the center of the robot. Through that portal, find a 9-pin DSUB socket into which you plug a common, "pass-through" serial cable to your client PC.⁵

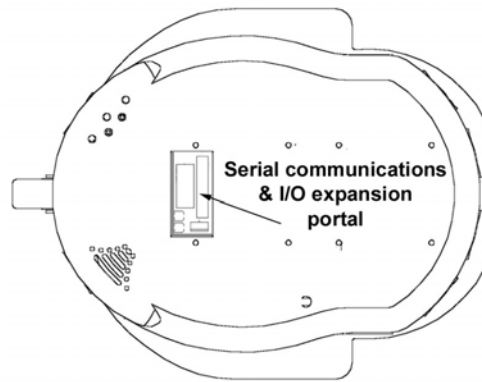


Figure 12. AmigoBot's expansion portal

Also inside that portal, find a 5-position latch-lock header that provides a second, auxiliary RS-232 serial port as well as 12 and 5 VDC power for accessories. Access the 40-position high-density IDC latching connector for its many I/O ports.

See Appendix A, *Ports and Connectors*, for details.

SONAR

The AmigoBot comes standard with a single array of eight sonar. The sonar positions are fixed: one on each side, four facing forward, and two at the rear, together providing nearly 360 degrees of range sensing. The sonar are numbered clockwise around the robot, beginning with the left side disc.

Sonar Rate and Sequence

The sonar array's transducers are multiplexed: Only one disc is active at a time. The sonar ranging acquisition rate is adjustable, normally set to 25 Hertz (40 milliseconds per transducer). Sensitivity ranges from 10 centimeters (six inches) up to five meters, depending on the ranging rate. You may

⁵ Since the ports are shared, power off or unplug the Serial Ethernet accessory or other attached device when using the `Host/System` port on top of AmigoBot.

control the sonar's firing pattern through software, too; the default is left-to-right in sequence 0 to 7 for each array. See the AmigoSH Chapters 5 and 6 for details.

Sonar Sensitivity

All eight sonar are controlled from a single board. Although calibrated at the factory, you may adjust the sonar sensitivity and range to accommodate differing AmigoBot operating environments. The sonar gain control is a one-turn screwcap accessible through a hole on the top and near the front of AmigoBot. You may have to remove an accessory to uncover the hole.

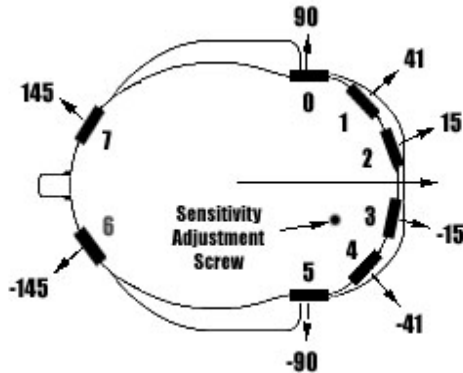


Figure 13. AmigoBot's sonar array

Using a flat-bladed screwdriver, turn the adjustment screw counterclockwise to make the sonar less sensitive to external noise and false echoes. Low sonar-gain settings reduce the robot's ability to see small objects. Under some circumstances, that is desirable. For instance, attenuate the sonar if you are operating in a noisy environment or on uneven or highly reflective floor—a heavy shag carpet, for example. If the sonar are too sensitive, they will “see” the carpet immediately ahead of the robot as an obstacle.

Increase the sensitivity of the sonar array by turning the gain-adjustment screw clockwise, making the sonar more likely to see small objects or objects at a greater distance. For instance, increase the sonar gain if you are operating in a relatively quiet and open environment with a smooth floor surface.

MOTORS, WHEELS AND ENCODERS

AmigoBot's tires are four inches (10.2cm) in diameter and made of soft, but firm rubber for good traction and low compressibility. AmigoBot's drive system uses high-speed, high-torque, reversible-DC motors. Each drive motor includes a high-resolution optical quadrature shaft encoder that provides 123 ticks per millimeter of wheel rotation for precise position and speed sensing and advanced dead-reckoning.

Three FLASH-based parameters influence AmigoBot's odometry: `RevCount` sets the differential encoder ticks for 180 degrees of rotation which AmigoSH uses to compute heading and speed of rotation. `TicksMM` is the number of ticks per millimeter of tire rotation which AmigoSH uses to compute speed and distance traveled. `DriftFactor`, a signed integer measured in 1/8192 units, gets added to or subtracted from the left-wheel encoder's value to compensate for unevenly sized tires.

See Chapters 5 and 6 for a detailed description of AmigoBot's odometry measures and controller-FLASH related variables.

SAFETY AMIGO SH WATCHDOGS

AmigoSH contains a FLASH-settable client `WatchDog` that will halt the robot's motion if communications between a PC client and the robot server are disrupted for a set time interval. The robot will automatically resume activity, including motion, as soon as communications are restored.

AmigoSH also contains a stall monitor. If the drive exerts a PWM pulse that equals or exceeds a configurable level (`StallVal`) and the wheels fail to turn, the motors relax (power off) for a configurable amount of time (`StallWait`). The server software also notifies the client which motor is stalled. When the `StallWait` time elapses, motor power automatically switches back on and motion continues under client control.

You may reconfigure these and many other FLASH-based parameter values to suit your application. See Chapter 6, *Updating & Reconfiguring AmigoSH*, for details.

Chapter 4 Quick Start and Self-Test

This chapter describes how to quickly setup, test, and operate your new *Team AmigoBot* with ARIA and with ARNetworking/MobileEyes, if you have the wireless Ethernet-to-serial device. For advanced robotics mapping, localization and navigation, please consult the *SonARNL* documentation.

PREPARATIVE ASSEMBLY

Your Team AmigoBot comes fully assembled, battery fully charged, and ready for out-of-the-box operation. You may need to attach some accessories that we shipped separately or detached for safety. Please consult the accessory manuals for attachment and setup details.

SELF TEST

The AmigoSH firmware in AmigoBot's controller contains a simple self-test routine that exercises your robot's drive motors. Place AmigoBot on the floor. Engage the self-test by pressing the black `Motors/Test` button once, then again anytime before five seconds elapse.⁶

During self test, AmigoBot turns left and right, forward and back, and then reverts to start up mode. If not, there's something wrong. Check the battery and be sure the tires are free to rotate.

CLIENT DEMONSTRATIONS

MOBILEROBOTS' robot-client software-development environments, including ARIA and ARNetworking, their respective demonstration programs `demo` and `serverDemo` (aka `ARIA Demo` and `ARNetworking Demo`, respectively), as well as the GUI networked-client application, `MobileEyes`, come on CD-ROM with your new robot. MOBILEROBOTS customers also may obtain these and related software and updates from our support website:

<http://robots.MobileRobots.com>

When installed, the software typically require around four megabytes of hard-disk space.

The Windows version of ARIA/ARNetworking is a self-extracting archive. Simply double-click the `ARIA... .exe` icon and follow the extraction program's instructions. Normally, the software is put into a directory named `C:\Program Files\MobileRobots\ARIA`. The demonstration programs get put into the `bin\` subdirectory. For convenience, you also may access them from the Start Menu's Programs option. The demonstration program sources, .NET® projects and workspace files are in their respective `examples\` and `ARNetworking\examples` subdirectories.

`MobileEyes` comes as a separate distribution archive, but also gets installed in the `C:\Program Files\MobileRobots\` path.

Linux users must have `superuser (root)` permissions in order to install ARIA with ARNetworking and `MobileEyes`. ARIA comes as an RPM installation archive:

```
rpm -ihv aria...
```

and gets put into `/usr/local/Aria`. The ARIA demonstration programs get put into the `bin/` subdirectory. The demonstration sources and `Makefiles` are in the `examples/` and `ARNetworking/examples` subdirectories.

Similarly, `rpm` extracts `MobileEyes` normally into `/usr/local/MobileEyes` and its executable is in the respective `bin/` subdirectory.

Linux users should also be sure they have permission to read/write through their PC's serial port that connects with the robot. The default is `/dev/ttyS0`. `ARIA demo` and `serverDemo` are terminal-based applications that do not include a GUI, so they do not require X-Windows. `MobileEyes`, on the other hand, does require X.

⁶ To avoid inadvertent engagement of the self-test, particularly for larger MobileRobots platforms, the controller cancels the self-test automatically after five seconds if you don't press the button a second time.

CLIENT-SERVER COMMUNICATIONS

Your AmigoBot requires a serial communication link with a client PC for operation:

- ✓ A tether cable from the robot's 9-pin serial connector inside the I/O portal
- ✓ A piggyback PC cabled to that same serial connector
- ✓ Wireless Ethernet-to-serial attached to the alternative Host connector underneath the robot
- ✓ Radio Modem attached to that connector underneath

First, please note well that you cannot connect with and run your AmigoBot with ARIA demo, ARNetworking serverDemo, or MobileEyes from a networked PC without special hardware: an Ethernet-to-serial accessory on the robot.

TETHERED WITH ARIA DEMO

Let's start by avoiding the networking issues for the moment and test the robot through a serial tether. Accordingly, unplug the power connector from your Ethernet-to-serial accessory on the robot if it is installed and attach a serial cable from your PC to the Host serial port through AmigoBot's access hole.

Windows users may select the ARIA demo from the Start menu, in the MobileRobots program group. Otherwise, start it from the ARIA bin\ directory.

Linux users will find the compiled demo in /usr/local/Aria/bin/ or in examples/. Start it:

```
% ./demo
```

Demo Startup Options

Table 1. ARIA demo and ArNetworking serverDemo command line arguments

-remoteHost <Host Name or IP> (abbreviated -rh)	Connect with robot through a remote host over the network instead of a serial port; requires special serial Ethernet hardware or IPTHRU software mediation.
-robotPort <Serial Port> (abbreviated -rp)	Connect with robot through specified serial port name; COM3, for example. COM1 or /dev/ttyS0 is the default.
-robotBaud <baudrate> (abbreviated -rb)	Connect with robot using the specified baudrate; 19200 or 38400, for example. Default is 9600.
-remoteRobotTcpPort <Number> (abbreviated -rrtp)	Remote TCP host-to-robot connection port number; default is 8101.

By default, the ARIA demo program connects with the robot through the serial port COM1 under Windows or /dev/ttyS0 under Linux. To change these and other connection options, either modify the source code (examples/demo.cpp) and recompile the program, or use a startup argument from the command line (Table 1).

For example, from the Windows Start:Run dialog, choose Browse... and select the ARIA demo program: C:\Program Files\MobileRobots\ARIA\bin\demo.exe. Then, type a command line argument at the end of the text in the Run dialog. To connect through the COM2 serial port on your PC, for example:

```
C:\Program Files\ActivMedia Robotics\ARIA\bin\demo.exe -rp COM2
```

A Successful Connection

ARIA prints out lots of diagnostic text as it negotiates a connection with the robot. If successful, the client requests various AmigoSH servers to start their activities, including sonar polling, position integration, and so on. The controller sounds an audible connection cue, and you should hear the robot's sonar ping with a distinctive and repetitive clicking. In addition, the motors-associated System LED on the top of AmigoBot should flash very fast (was flashing slowly while awaiting connection). Note that the ARIA demo automatically engages your robot's motors through a special client command. Normally, the motors are disengaged when first connecting.

The amber User LED on AmigoBot should blink, too, indicating ARIA-client to AmigoSH-server communications, too.

Table 2. ARIA demo operation modes

MODE	HOT KEY	DESCRIPTION
laser	l	Displays the closest and furthest readings from the robot's laser range finder
io	i	Displays the state of the robot's digital and analog-to-digital I/O ports
position	p	Displays the coordinates of the robot's position relative to its starting location
bumps	b	Displays the status of the robot's bumpers
sonar	s	Displays the robot's sonar readings
camera	c	Controls and exercises the robot's pan-tilt-zoom robotic camera
gripper	g	Controls, exercises, and displays status of the robot's Gripper
wander	w	Sends the robot to move around at its own whim, while avoiding obstacles
teleop	t	Drive and steer the robot via the keyboard or a joystick; avoids collisions
unguarded	u	Same as teleop, except no collision avoidance
direct	d	AmigoSH-direct command mode

Operating the ARIA Demo

When connected with the ARIA demo client, your robot becomes responsive and intelligent. For example, it moves cautiously: Although it may drive toward an obstacle, your AmigoBot will not crash because the ARIA demo includes obstacle-avoidance behaviors which enable the robot to detect and actively avoid collisions.

Table 2. Keyboard teleoperation

KEY	ACTION
↑	forward
↓	reverse
←	turn left
→	turn right
space bar	all stop

The ARIA demo also displays a menu of robot operation options. The default mode of operation is teleop. In teleop mode, you drive the robot manually, using the arrow keys on your keyboard or a joystick connected to the client PC's joystick port.

While driving from the keyboard, hold down the respective keys to simultaneously drive the robot forward or backward and turn right or left. For instance, hold down the up-arrow key to have the robot accelerate forward to its cruising speed of

around 400 millimeters per second (defined in the source code). Press the space bar to have the robot slow down and stop. Press and hold the right- or left-arrow key to have the robot rotate or turn in an arc if you also hold down the up- or down-arrow key.

The other modes of ARIA demo operation give you access to your robot's various sensors and accessories, including encoders, sonar, a pan-tilt-zoom robotic camera, I/O port states and so on, although not all accessories are or can be attached to your AmigoBot. Accordingly, use the ARIA demo not only as a demonstration tool, but as a diagnostic one, as well, if you suspect a sensor or effector has failed or is working poorly. The demo also is useful for calibrating your robot's drive system (position mode).

Access each ARIA demo mode by pressing its related hot-key: 't', for instance, to select teleoperation. Each mode includes onscreen instructions and may have sub-menus for operating of the respective device.

Disconnecting

When you finish, press the `Esc` key to disconnect the ARIA client from your AmigoBot and exit the ARIA demonstration program. Your AmigoBot should disengage its drive motors and stop moving, and its sonar should stop firing. You may now slide the robot's `Power` switch to `OFF`.

NETWORKING WITH MOBILEEYES

If you have the wireless Ethernet-to-serial accessory for your Team AmigoBot, you may operate the robot with the ARIA demo program over your local TCP/IP-based network. Simply start the program with the `-remoteHost` (or `-rh`) command line option and specify AmigoBot's Ethernet-to-serial device's IP address or related hostname. The system requires that you also have a radio-Ethernet access point nearby or have equipped a workstation or laptop with wireless Ethernet and have set up a peer-to-peer link with your AmigoBot's Ethernet-to-serial device.⁷

To use the MobileEyes GUI client for much more advanced, network-based robotics control, you need to do things a bit differently. You still need the Ethernet-to-serial device on the robot. But, instead of ARIA demo, you need ARNetworking's `serverDemo` to mediate communications between MobileEyes and AmigoBot.

Start serverDemo

The ARNetworking client, `serverDemo`, works to mediate communications between Team AmigoBot over the network and remote clients, such as the GUI MobileEyes. To examine its inner workings, look at `serverDemo.cpp` (and others) sources in the `ARNetworking/examples` subdirectory.

Start `serverDemo` just like `demo` using the `-remoteHost` argument to specify AmigoBot's IP address or hostname. `ServerDemo` can be run anywhere on the network from which your MobileEyes and your AmigoBot have access, such as on the same PC as you run MobileEyes.

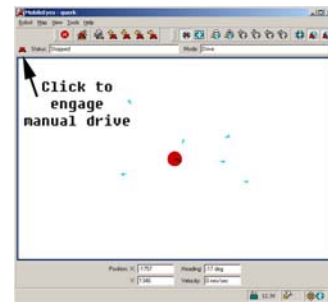


Figure 14. Simple serverDemo-mediated connection between AmigoBot and MobileEyes. MobileEyes is a terrific GUI-based robotics command and control client, but works best when running AmigoBot with SONARNL in a mapped space.

Connect MobileEyes with serverDemo, not AmigoBot directly.

Start MobileEyes and Connect with serverDemo

From Windows, simply double-click the `MobileEyes.exe` program normally located in the `C:\Program Files\MobileRobots\MobileEyes\bin` directory. With Linux, you need to have started X and, from a terminal session, navigate to the `MobileEyes/bin` directory in `/usr/local` and execute it:

```
$ ./MobileEyes &
```

In the MobileEyes startup dialog, enter the hostname or IP address of the PC on which you are running `serverDemo`, **not the IP address** of the AmigoBot itself.

⁷ Wireless networking details are beyond the scope of this manual. Your IT department is the first place to start. Let them know that the robot needs a static IP address and read/write access through port 8101. Contact MOBILEROBOTS support for additional help.

Operating MobileEyes

MobileEyes is not that interesting when run with serverDemo. Simply click the robot icon on the menu bar to engage manual drive and operate the robot as you did with the ARIA demo. Your Team AmigoBot with MobileEyes really comes to life when used with SonARNL for map-based path-planning and navigation. Nonetheless, this demonstration presents the networking, control and programming concepts inherent in ARNetworking.

TROUBLESHOOTING

Most problems occur when attempting to connect `demo` or `serverDemo` with your AmigoBot for the first time. The process can be daunting if you don't make the right connections and installations.

The most common mistake is not having your robot's "parameter" file located in the `ARIA/params` directory. For AmigoBot, the ARIA-parameter file's name is `amigo-sh.p`. It and all other MOBILEROBOTS platform ARIA-based parameter files either come with the latest version of ARIA with ARNetworking or can be retrieved from the <http://robots.MobileRobots.com> support website.

Another common mistake with Linux is not having the proper permissions on the connecting serial port.

Make sure your robot's batteries are fully charged. The robot servers shut down and won't allow a connection at under `ShutdownVolts`.

ATTENTION!

The demo/serverDemo-to-robot connection is *SERIAL* only. Accordingly, run them over radio modems or over the network with a wireless Ethernet-to-serial device.

If you are using an Ethernet-to-serial device, its serial connection is internal and established at the factory; you should not have problems with those cables. Simply make sure the device has power and remove any serial cable that is plugged into the other Host serial port on the top of the robot as it may interfere with internal serial communication.

With other serial connections, make sure to use the proper cable: a "pass-through" one, minimally connecting pins 2, 3, and 5 of your PC's serial port to their respective contacts of the robot's serial port on the User Control Panel.

If you access the wrong serial port, the ARIA demonstration program will display an error message. If the robot server isn't listening or if the serial link is severed somewhere between the client and server (cable loose or the radio is off, for instance), the client will attempt "Syncing 0" several times and fail. In that case, `Reset` AmigoBot and check your serial connections.

If for some reason communications get severed between the ARIA client and the robot, but both the client and server remain active, you may revive the connection with little effort: If you are using wireless communications, first check and see if the robot is out of range.

Communications also will fail if the client and/or server is somehow disabled during a session. For instance, if you inadvertently switch off the robot's `Power` or press the `Reset` button, you must restart the connection. Turning the `Power` switch `OFF` and then back `ON`, or pressing the `Reset` button puts the robot servers back to their wait state, ready to accept client connections again. If the ARIA demo or other client application is still active, simply press `esc` and restart.

Chapter 5 AmigoSH

All MOBILEROBOTS platforms use a client-server mobile robot-control architecture. In the model, the robot's controller servers work to manage all the low-level details of the mobile robot's systems. These include operating the motors, firing the sonar, collecting and reporting sonar and wheel encoder data, and so on—all on command from and reporting to a separate client application, such as the ARIA demo.

With this client/server architecture, robotics applications developers do not need to know many details about a particular robot server, because the client insulates them from this lowest level of control. Some of you, however, may want to write your own robotics control and reactive planning programs, or just would like to have a closer programming relationship with your robot. This chapter explains how to communicate with and control your AmigoBot via its AmigoBot-SH Operations Software (AmigoSH) client-server interface. The same AmigoSH functions and commands are supported in the various client-programming environments that accompany your robot or are available for separate license.

Experienced users can be assured that AmigoSH is upwardly compatible with all MOBILEROBOTS platforms, implementing the same commands and information packets that first appeared in the Pioneer 1-based PSOS, in the original Pioneer 2-based P2OS, and more recent AROS and ARCOS-based Pioneer 2s and 3s, as well as Performance PeopleBot, PatrolBot and PowerBot. AmigoSH, of course, extends the servers to add new functionality, improve performance, and provide additional information about the robot's state and sensing.

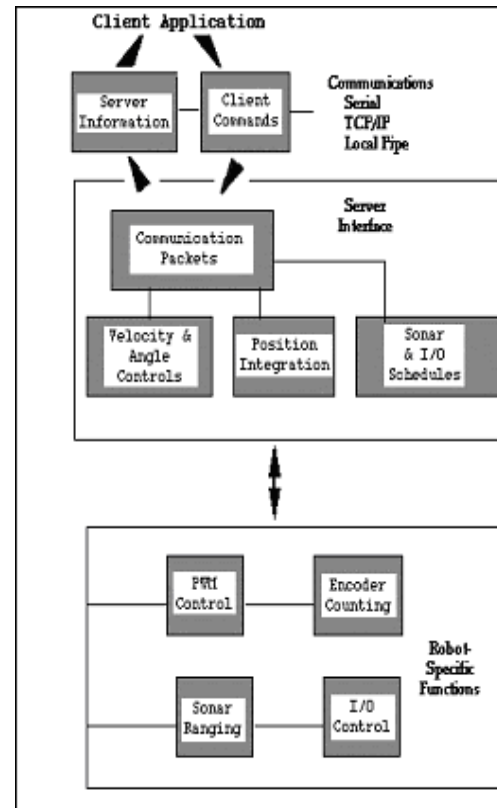


Figure 15. ActivMedia Robotics' client-server control architecture

CLIENT-SERVER COMMUNICATION PACKET PROTOCOLS

MOBILEROBOTS platforms communicate with a client using special client-server communication packet protocols, one for command packets from client to server and another for Server Information Packets (SIPs) from the server to client. Both protocols are bit streams consisting of five main elements: a two-byte header, a one-byte count of the number of subsequent packet bytes, the client command or SIP packet type, command data types and argument values or SIP data bytes, and, finally, a two-byte checksum. Packets are limited to a maximum of 207 bytes each.

The two-byte header which signals the start of a packet is the same for both client-command packets and SIPs: 0xFA (250) followed by 0xFB (251). The subsequent count byte is the number of all *subsequent* bytes in the packet including the checksum, but not including the byte count value itself or the header bytes.

Data types are simple and depend on the element (see descriptions below): client commands, SIP types, and so on, are single 8-bit bytes, for example. Command arguments and SIP values may be 2-byte integers, ordered as least-significant byte first. Some data are strings of up to a maximum 200 bytes, prefaced by a length byte. Unlike common data integers, the two-byte checksum appears with its *most-significant* byte first.

Packet Checksum

Calculate the client-server packet checksum by successively adding data byte pairs (high byte first) to a running checksum (initially zero), disregarding sign and overflow. If there are an odd number of data bytes, the last byte is XORed to the low-order byte of the checksum.

```
AREXPORT ArTypes::Byte2 ArRobotPacket::calcChecksum(void)
{
    int i;
    unsigned char n;
    int c = 0;

    i = 3;
    n = myBuf[2] - 2;
    while (n > 1) {
        c += ((unsigned char)myBuf[i]<<8) | (unsigned char)myBuf[i+1];
        c = c & 0xffff;
        n -= 2;
        i += 2;
    }
    if (n > 0)
        c = c ^ (int)((unsigned char) myBuf[i]);
    return c;
}
(from MobileRobots ARIA ArRobotPacket.cpp)
```

NOTE: The checksum integer is placed at the end of the packet, with its bytes in the reverse order of that used for data; that is, b_0 is the high byte and b_1 is the low byte.

Packet Errors

AmigoSH ignores a client command packet whose byte count exceeds 204 (total packet size of 207 bytes) or has an erroneous checksum. The client should similarly ignore erroneous SIPs.

Because of the real-time nature of client-server mobile-robotics interactions, we made a conscious decision to provide an unacknowledged communication packet interface. Retransmitting server information or command packets typically serves no useful purpose because old data is useless in maintaining responsive robot behaviors.

Nonetheless, the client-server interface provides a simple means for dealing with ignored command packets: Most of the client commands alter state variables in the server. By examining those values in respective SIPs, client software may detect ignored commands and re-issue them until achieving the correct state.

Table 3. Client command packet protocol

COMPONENT	BYTES	VALUE	DESCRIPTION
header	2	0xFA, 0xFB	Packet header; same for client and server
byte count	1	N	Number of command/argument bytes plus Checksum's two bytes, but not including Byte Count itself or the header bytes. Maximum of 249.
command number	1	0 - 255	Client command number; see Table 5.
argument type	1	0x3B or 0x1B or 0x2B	Required data type of command argument: positive integer, negative or absolute integer, or string
argument	n	data	Command argument; always 2-byte integer or string containing length prefix
checksum	2	computed	Packet integrity checksum

THE CLIENT-SERVER CONNECTION

Before exerting any control, a client application must first establish a connection with the robot server via a serial link through the robot microcontroller's `HOST` serial port either via the internal `HOST` or the User Control Panel `SERIAL` connector. After establishing the communication link, the client then sends commands to and receives operating information from the server.

When first started or reset, AmigoSH is in a special wait state listening for communication packets to establish a client-server connection.⁸ To establish a connection, the client application must send a series of three synchronization packets containing the `SYNC0`, `SYNC1` and `SYNC2` commands in succession, and retrieve the server responses.

Specifically, and as examples of the client command protocol described below, the sequence of synchronization bytes is:

```

SYNC0: 250, 251, 3, 0, 0, 0
SYNC1: 250, 251, 3, 1, 0, 1
SYNC2: 250, 251, 3, 2, 0, 2

```

When in wait mode, AmigoSH echoes the packets verbatim back to the client. The client should listen for the returned packets and only issue the next synchronization packet after it has received the appropriate echo.

Autoconfiguration (SYNC2)

AmigoSH automatically sends robot identifying information back to the client following the last synchronization packet (`SYNC2`). The configuration values are three NULL-terminated strings that comprise the robot's FLASH-stored `name`, `class`, and `subclass`. You may uniquely `name` your robot with the FLASH configuration tool we provide. The `class` and `subclass` are constants normally set at the factory and not changed thereafter. (See next chapter for details.)

The `class` string typically is `Pioneer`. The `subclass` depends on your robot model; Amigo-SH for Team AmigoBot. Clients may use these identifying strings to self-configure their own operating parameters. ARIA, for instance, loads and uses the robot's related parameter files found in the special `Aria/params` directory.

Opening the Servers—OPEN

Once you've established a connection with AmigoSH, your client should send the `OPEN` command #1 (250, 251, 3, 1, 0, 1) to the server, which causes the microcontroller to perform a few housekeeping functions, start its various servers, such as for the sonar and motors, and begin transmitting server information to the client.

Note that when at first connected, your robot's motors are disabled regardless of their state when last connected. To enable the motors after starting a connection, you must either do it manually (press the white `MOTORS` button on the User Control Panel) or have your client send an `ENABLE` client command #4 with an integer argument of 1. See *Client Commands* below.

**Once connected, send the `ENABLE` command
or press the `MOTORS` button on the User Control Panel
to enable your robot's motors.**

Server Information Packets

Once `OPENED`, AmigoSH automatically and repeatedly sends a packet of information over the `HOST` serial port back to the connected client. The standard AmigoSH SIP informs the client about a number of operating states and readings, using the order and data types described in the nearby Table. AmigoSH also supports several additional SIP types. See following sections for details.

⁸ There also is maintenance mode for ARCOS downloads and parameter updates; see next chapter for details.

Table 4. AmigoSH standard SIP contents

LABEL	DATA	DESCRIPTION
HEADER	2 bytes	Exactly in order 0xFA (250), 0xFB (251)
BYTE COUNT	byte	Number of data bytes + 2 (checksum), not including header or byte-count bytes
TYPE	0x3s	Motors status; s = 2 when motors stopped or 3 when robot moving.
XPOS	int	Wheel-encoder integrated coordinates in millimeters (DistConvFactor‡ = 1.0).
YPOS	int	
THPOS	int	Orientation in degrees (AngleConvFactor‡ = 1.0).
L VEL	int	Wheel velocities in millimeters per second (VelConvFactor‡ = 1.0)
R VEL	int	
BATTERY	byte	Battery charge in tenths of volts (101 = 10.1 volts, for example)
STALL AND BUMPERS	uint‡	Motor stall and bumper indicators. Bit 0 is the left wheel stall indicator, set to 1 if stalled. Bits 1-7 correspond to the first bumper I/O digital input states (accessory dependent). Bit 8 is the right wheel stall, and bits 9-15 correspond the second bumper I/O states, also accessory and application dependent.
CONTROL	int	Setpoint of the server's angular position servo in degrees
FLAGS	uint	Bit 0 motors status; bits 1-4 sonar array status; bits 5,6 STOP; bits 7,8 ledge-sense IRs; bit 9 joystick fire button; bit 10 auto-charger power-good.
COMPASS	byte	Electronic compass accessory heading in 2-degree units
SONAR COUNT	byte	Number of new sonar readings included in SIP
NUMBER	byte	If Sonar Count>0, is sonar disc number 0-31; readings follow
RANGE	uint	Corrected sonar range value in millimeters (RangeConvFactor‡ = 1.0)
...REST OF THE SONAR READINGS...		
GRIP STATE	byte	Gripper state byte.
ANPORT	byte	Selected analog port number 1-5
ANALOG	byte	User Analog input (0-255=0-5 VDC) reading on selected port
DIGIN	byte	Byte-encoded User I/O digital input
DIGOUT	byte	Byte-encoded User I/O digital output
BATTERYX10	int	Actual battery voltage in 0.1 V (especially useful for battery voltages > 25.5)
CHARGE STATE	byte	Automated recharging state byte; -1 = unknown; 0=not charging; 1=bulk; 2=overcharge; 3=float.
CHECKSUM	int	Packet-integrity checksum

‡ Client-side data-conversion factor. Consult the ARIA parameter file your robot.

‡ Explicitly, an unsigned integer; all others sign-extended

Keeping the Beat—PULSE

An AmigoSH safety watchdog expects that, once connected, your robot receives at least one client command packet from the attached PC every `watchDog` seconds, as defined in your robot's FLASH (default is two seconds). Otherwise, it assumes the client-server connection is broken and stops the robot.

Some clients—ARIA-based ones, for instance—use the good practice of sending a `PULSE` client command #0 (250, 251, 3, 0, 0, 0) just after `OPEN`. And if your client application will be otherwise distracted for some time, periodically issue the `PULSE` command to let your robot server know that your client is indeed alive and well. It has no other effect.

If the robot shuts down due to lack of communication with the client, it will revive upon receipt of a client command and automatically accelerate to the last-specified speed and heading setpoints.

Closing the Connection—CLOSE

To close the client-server connection, which automatically disables the motors and other functions like sonar, simply issue the client `CLOSE` command #2 (250, 251, 3, 2, 0, 2).

Most of AmigoSH' operating parameters return to their FLASH-based default values upon disconnection with the client.⁹

CLIENT COMMANDS

AmigoSH has a structured command format for receiving and responding to directions from a client for control and operation of your robot. Client commands are comprised of a one-byte command number optionally followed, if required by the command, by a one-byte description of the argument type and then the argument value.

The number of client commands you may send per second depends on the `HOST` serial baud rate, average number of data bytes per command, synchronicity of the communication link, and so on. AmigoSH' command processor runs on a one millisecond interrupt cycle, but the server response speed depends on the command. Typically, limit client commands to a maximum of one every 3-5 milliseconds or be prepared to recover from lost commands.

Table 5. AmigoSH client-side command set¹⁰

COMMAND	#	ARGS	DESCRIPTION	VERSION
			Before Client Connection	
SYNC0	0	none	Start connection. Send in sequence. AmigoSH echoes synchronization commands back to client, and robot-specific auto-synchronization after SYNC2.	1.0
SYNC1	1	none		
SYNC2	2	none		
			After Established Connection	
PULSE	0	none	Reset server watchdog.	1.0
OPEN	1	none	Start up servers.	1.0
CLOSE	2	none	Close servers and client connection.	1.0
POLLING	3	str	Change sonar polling sequence.	1.0
ENABLE	4	int	1=enable; 0=disable the motors.	1.0
SETA	5	int	Set translation acceleration, if positive, or deceleration, if negative; in mm/sec ² .	1.0
SETV	6	int	Set maximum/move translation velocity; mm/sec.	1.0
SETO	7	none	Reset local position to 0,0,0 origin.	1.0
MOVE	8	int	Translate (+) forward or (-) back mm distance at SETV speed	1.0
ROTATE	9	int	Rotate (+) counter- or (-) clockwise degrees/sec.	1.0
SETRV	10	int	Sets maximum/turn rotation velocity; degrees/sec.	1.0
VEL	11	int	Translate at mm/sec forward (+) or backward (-) (SETV limit).	1.0
HEAD	12	int	Turn at SETRV speed to absolute heading; ±degrees (+ = ccw).	1.0
DHEAD	13	int	Turn at SETRV speed relative to current heading; (+) counter- or (-) clockwise degrees.	1.0
SAY	15	str	Play up to 20 duration, tone sound pairs through User Control Panel piezo speaker.	1.0
JOYREQUEST	17	int	Request one or continuous stream (>1) or stop (0) joystick SIPs	1.3
CONFIG	18	none	Request a configuration SIP.	1.0
ENCODER	19	int	Request one, a continuous stream (>1), or stop (0) encoder SIPs.	1.0
RVEL	21	int	Rotate robot at (+) counter- or (-) clockwise; degrees/sec (SETRV limit).	1.0
DCHEAD	22	int	Adjust heading relative to last setpoint; ± degrees (+ = ccw)	1.0
SETRA	23	int	Change rotation de(-) or (+) acceleration, in degrees/sec ²	1.0
SONAR	28	int	1=enable, 0=disable all the sonar; otherwise, use bit 0 to enable (1) or disable (0) a particular array 1-4, as specified in argument bits 1-4.	1.0

⁹ With earlier OSes, the changes persisted between sessions and reverted to the FLASH defaults only after the controller was reset.

¹⁰ To no harm, not all client commands work with AmigoBot, depending on accessories and options. For example, there is only one auxiliary serial port, so GETAUX2 and GETAUX3 get ignored by AmigoSH.

STOP	29	none	Stop the robot; motors remain enabled	1.0
DIGOUT	30	2 byte	Set (1) or reset (0) User Output ports. Bits 8-15 is a byte mask that selects, if set (1), the output port(s) for change; Bits 0-7 set (1) or reset (0) the selected port(s).	1.0
VEL2	32	2 byte	Set independent wheel velocities; bits 0-7 for right wheel, bits 8-15 for left wheel; in 20mm/sec increments.	1.1
GRIPPER	33	int	Gripper server commands. See the Gripper or PeopleBot Manual for details.	1.0
ADSEL	35	int	Selects the A/D port number for reporting ANPORT value in standard SIP.	1.0
GRIPPERVAL	36	int	Gripper server values. See Gripper or PeopleBot Manual for details.	1.0
GRIPREQUEST	37	int	Request one, a continuous stream (>1), or stop (0) Gripper SIPs.	1.0
IOREQUEST	40	int	Request one (1), a continuous stream (>1), or stop (0) IO SIPs.	1.0
TTY2	42	str	Sends string argument to serial device connected to AUX1 serial port.	1.0
GETAUX	43	uint	Request to retrieve 1-200 bytes from the AUX1 serial port; 0 flushes the buffer.	1.0
BUMPSTALL	44	int	Stall robot if no (0), only front (1) while moving forward, only rear (2) while moving backward, or either (3) bumpers contacted when robot moving in related direction.	1.0
TCM2	45	int	TCM2 module commands; see TCM2 Manual for details.	1.0
JOYDRIVE	47	int	1=allow joystick drive from port while connected with a client; 0 (default) disallows.	1.0
SONARCYCLE	48	uint	Change the sonar cycle time; in milliseconds.	1.0
HOSTBAUD	50	int	Change the HOST serial port baud rate to 0=9600, 1=19200, 2=38400, 3=57600, or 4=115200.	1.0
AUX1BAUD	51	int	Change the AUX1 serial port baud rate (see HOSTBAUD).	1.0
AUX2BAUD	52	int	Change the AUX2 serial port baud rate (see HOSTBAUD).	1.0
AUX3BAUD	53	int	Change the AUX3 serial port baud rate (see HOSTBAUD).	1.0
E_STOP	55	none	Emergency stop; very abrupt by overriding deceleration.	1.0
M_STALL	56	int	Argument 1=MOTORS button off causes a stall	1.0
GYROREQUEST	58	int	Request one, a continuous stream (>1), or stop (0) Gyro SIPs.	1.0
LCDWRITE	59	str	Display a message on the LCD accessory: byte 1=starting column (1-19); byte 2=starting row (1-4); byte 3=1 if clear line contents first, otherwise 0; bytes 4- = up to 20 chars, NULL-terminated.	1.3
TTY4	60	str	Send string argument out to device connected at AUX3 serial port.	1.0
GETAUX3	61	int	Request to retrieve 1-200 bytes from the device connected at the AUX3 serial port; 0 flushes the buffer.	1.0
TTY3	66	str	Send string argument out to device connected at AUX2 serial port.	1.0
GETAUX2	67	int	Request to retrieve 1-200 bytes from the device connected at the AUX2 serial port; 0 flushes the buffer.	1.0
CHARGE	68	int	0=release; 1=deploy autocharge-docking mechanism.	1.0
ARM	70-80	int	Pioneer Arm-related commands. See Arm manual for details.	1.1
ROTKP	82	int	Change working rotation Proportional PID value.	1.0
ROTKV	83	int	Change working rotation Derivative PID value.	1.0
ROTKI	84	int	Change working rotation Integral PID value.	1.0
TRANSP	85	int	Change working translation Proportional PID value.	1.0
TRANSPV	86	int	Change working translation Derivative PID value.	1.0
TRANSKI	87	int	Change working translation Integral PID value.	1.0
REVCOUNT	88	int	Change working differential encoder count.	1.1
DRIFTFACTOR	89	int	Change working drift factor.	1.0

SOUNDTOG	92	int	0=mute User Control piezo; 1 = enable.	1.0
TICKSMM	93	int	Change working encoder ticks per millimeter tire travel.	1.1
BATTEST	250	int	Artificially set the battery voltage; argument in tens volts (100=10V); 0 to revert to real voltage	1.0
RESET	253	none	Force a power on-like reset of the microcontroller.	1.0
MAINTENANCE	255	none	Engage microcontroller maintenance (ARSHstub) mode.	1.0

Client Motion Commands

The AmigoSH motor-control servers accept several different motion commands of two types: either independent-wheel (VEL2) or translation/rotation movements (VEL, ROT, etc). Actually, VEL2 commands are recomposed at the server into their translation and rotation components and the corresponding limits and (de)accelerations get applied. The AmigoSH servers automatically abandon any translation or rotation setpoints and switch to independent wheel velocity- or translation/rotation-type controls when your client issues a command of the opposite type.

Table 6. Client motion-related commands

ROTATION	
head (#12)	Turn to absolute heading at SETRV max velocity
dhead (#13), dhead (#22)	Turn to heading relative to control point at SETRV max velocity
rotate (#9)	Rotate at SETRV velocity
rvel (#21)	Rotate at + (counter) or - (clockwise) deg/sec up to SETRV max
TRANSLATION	
vel (#11)	Translate forward/reverse at prescribed velocity (SETV maximum)
move (#8)	Translate distance at SETV max velocity
INDEPENDENT	WHEEL
vel2 (#32)	Set velocity for each side of robot

Note that once connected, the robots' motors are *disabled*, regardless of their state when last connected. Accordingly, you must either enable the motors manually (white MOTORS button on the User Control Panel) or send the motors ENABLE client command #4 with the argument value of one.¹¹ Monitor the status of the motors with bit 0 of the Flags integer in the standard SIP.

ROBOTS IN MOTION

When AmigoSH receives a motion command, it accelerates or decelerates the robot at the translation SETA (command #5) and rotation SETRA (command #23) rates until the platform either achieves its SETV (command #6) maximum translation and SETRV (command #10) maximum rotation speeds (or VEL2 equivalents), or nears its goal. Accordingly, rotation headings and translation setpoints are achieved by a trapezoidal velocity function, which AmigoSH recomputes each time it receives a new motion command.¹²

AmigoSH automatically limits VEL2-, VEL-, and RVEL-specified velocities to previously imposed, client-modifiable SETVEL and SETRV maximums, and ultimately by absolute, platform-dependent, FLASH-embedded constants (TOP values). Similarly, the distinct acceleration and deceleration parameters for both translation and rotation are limited by FLASH constants. AmigoSH initializes these values

¹¹ Alternatively, disable the motors with the ENABLE command argument of zero.

¹² Note that acceleration and deceleration are distinct values, settable via SETA for translation and SETRA for rotation.

upon microcontroller startup or reset from related FLASH parameters. The speed limits, either from FLASH or when changed by SETV or SETRV commands, take effect on subsequent commands, not on current translation or rotation. The maximums revert to their FLASH defaults when disconnected from a client.

The orientation commands HEAD (#12), DHEAD (#13), DCHEAD (#22) turn the robot with respect to its internal dead-reckoned angle to an absolute heading (0-359 degrees), relative to its immediate heading, or relative to its current heading setpoint (achieved or last commanded heading), respectively. In general, positive relative heading command arguments turn the robot in a counterclockwise direction. However, the robot always turns in the direction that will achieve its heading most efficiently. Accordingly, relative-heading arguments greater than 179 degrees automatically get reduced to 179 or less degrees with a concomitant change in direction of rotation.

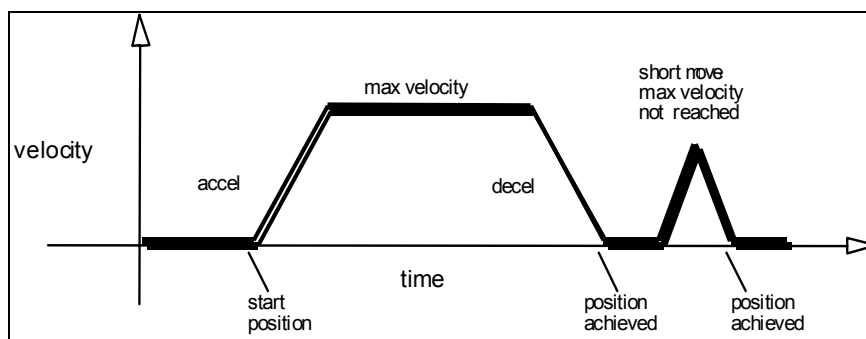


Figure 16. Trapezoidal velocity profile

The E_STOP command #55 or the STOP button that is found on some MOBILEROBOTS platforms override deceleration and abruptly stop the robot in the shortest distance and time possible. Accordingly, the robot brakes to zero translation and rotation velocities with very high deceleration and remains stopped until it receives a subsequent translation or rotation velocity command from the client or until the STOP button is reset. (See E_STOP and E_STALL later in this chapter.)

PID Controls

The AmigoSH drive servers use a common Proportional-Integral-Derivative (PID) system with wheel-encoder feedback to adjust a pulse-width-modulated (PWM) signal at the motor drivers to control the power to the motors. The motor-duty cycle is 50 microseconds (20 KHz); pulse-width is proportional 0-500 for 0-100% of the duty cycle. The AmigoSH drive servers recalculate and adjust your robot's trajectory and speed every five milliseconds.

The PID values for translation and rotation and maximum PWM are reconfigurable FLASH parameters in your robot's microcontroller. You also may temporarily update the PID values with the AmigoSH client commands #84 through #87. On-the-fly changes persist until the client disconnects. Translation PID values apply to independent wheel-velocity mode, as well.

The P-term value K_p increases the overall gain of the system by amplifying the position error. Large gains will have a tendency to overshoot the velocity goal; small gains will limit the overshoot but cause the system to become sluggish. We've found that a fully loaded robot works best with a K_p setting of around 15 to 30, whereas a lightly loaded robot may work best with K_p in the range of 20 to 50.

The D-term K_v provides a PID gain factor that is proportional to the output velocity. It has the greatest effect on system damping and minimizing oscillations within the drive system. The term usually is the first to be adjusted if you encounter unsatisfactory drive response. Typically, we find K_v to work best in the range of 10 to 30 for lightly to heavily loaded robots, respectively. If your robot starts to vibrate or shutter, reduce K_v .

The I-Term K_i moderates any steady state errors thereby limiting velocity fluctuations during the course of a move. At rest, your robot will seek to "zero out" any command position error. Too large of a K_i factor will cause an excessive windup of the motor when the load changes, such as when climbing

over a bump or accelerating to a new speed. Consequently, we typically use a minimum value for K_i in the range of 0 to 10 for lightly to heavily loaded robots respectively.

Position Integration

MOBILEROBOTS platforms track their position and orientation based on dead-reckoning from wheel motion derived from encoder readings. The AmigoSH-based robot maintains its internal coordinate position in platform-dependent units, but reports the values in platform-independent millimeters and degrees in the standard SIP (Xpos, Ypos, and Thpos).

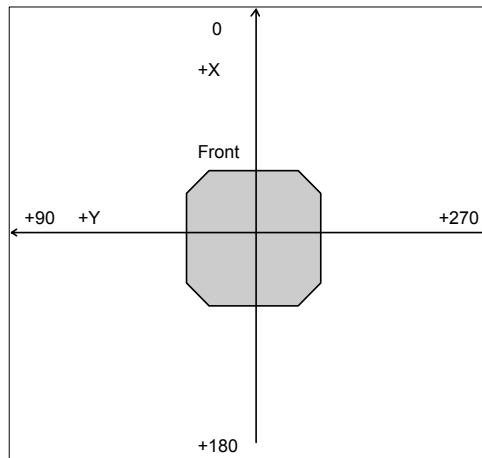


Figure 17. Internal coordinate system

Be aware that registration between external and internal coordinates deteriorates rapidly with movement due to gearbox play, wheel imbalance and slippage, and many other real-world factors. You can rely on the dead-reckoning ability of the robot for just a short range—on the order of a few meters and one or two revolutions, depending on the surface. Carpets tend to be worse than hard floors.

Also, moving either too fast or too slow tends to exacerbate the absolute position errors. Accordingly, consider the robot's dead-reckoning capability as a means of tying together sensor readings taken over a short period of time, not as a method of keeping the robot on course with respect to a global map.

On start-up, the robot is at the origin (0, 0, 0), pointing toward the positive X-axis at 0 degrees. Absolute angles vary between 0 and 359.

You may reset the internal coordinates back to 0,0,0 with the `SET0` command #7.

DriftFactor, RevCount and TicksMM

Three client commands let you change, albeit momentarily for the current client-server connection, those values that affect translation, rotation and drift in your robot. `TicksMM` is the number of encoder ticks per millimeter tire rotation for translation speed and distance computations. The default FLASH value can be changed on-the-fly during a client connection session with the `TICKSMM` client command #93 and unsigned integer value.

`DriftFactor` is a signed value in 1/8192 increments that gets added to or subtracted from the left wheel encoder's ticks to correct for tire circumference differences and consequent translation and rotation drift. `DriftFactor` defaults to its FLASH value on start up or reset, and can be changed on-the-fly with the `DRIFTFACTOR` client command #89 with signed integer argument.

The `RevCount` parameter is the differential number of encoder ticks for a 180-degree rotation of the robot and is used to compute and execute headings. Like `DriftFactor` and `TicksMM`, `RevCount` defaults to its FLASH value on startup or reset, and can be changed on-the-fly with the `REVCOUNT` client command #88 and unsigned integer argument.

SONAR

When connected with and opened by the client, AmigoSH automatically begins firing AmigoBot's 8-disc sonar array, one disc at a time, as initially sequenced and enabled in your robot's FLASH parameters. The sonar servers also begin sending the sonar-ranging results to the client via the standard SIP.

Enable/Disabling Sonar

Use the `SONAR` client command #28 with argument value of one or zero to enable or disable AmigoBot's sonar array.

Polling Sequence

AmigoBot's sonar array polls each disc at a rate and in the sequence defined in your robot microcontroller's FLASH parameters. (Consult the next chapter on how to change the FLASH settings.) Use the sonar `POLLING` command #3 to have your client change the firing sequence, and the `SONAR_CYCLE` command #48 to change the rate. The changes persist until you restart the client-server connection.

The `POLLING` command string argument consists of a sequence of sonar numbers one through eight. You may include up to 16 sonar numbers in the sequence, and you may repeat a sonar number two or more times in a sequence. If a sonar number does not appear in an otherwise altered sequence, the disc will not fire. If you do repeat a sonar in the sequence, know that ARIA and related clients ignore the first reading if two sonar ranging data appear in the same SIP.

For compatibility with earlier robot operating systems, if the string is empty, all the sonar in the array get disabled, but their polling sequence remains unaltered, just as if you had sent the `SONAR` command with an argument value of zero.

Polling Rate

For earlier Pioneer microcontroller versions, the sonar polling rate was fixed: one sonar disc per array got polled every 40 milliseconds. So, for an eight-disc array with a sequential polling sequence (12345678 by default), any one sonar transducer would be read every 320 milliseconds. That common cycle timing accommodates ranging out to the maximum of the sonar of over six meters for general applications, including features recognition and localization. For other applications, such as close-in obstacle avoidance, a shorter range but faster rate of update is better.

Use the `SONAR_CYCLE` client command #48 to change the cycle timing on the fly to the command integer's argument value in milliseconds. Minimum and maximum values are two and 120 milliseconds, respectively. The default value is set in FLASH, normally to the legacy 40 milliseconds.

STALLS AND EMERGENCIES

In an emergency, your client may want the robot to stop quickly, not subject to normal deceleration. In that case, send the `E_STOP` command (#55).

ACCESSORY COMMANDS AND PACKETS

Several types of alternative server information packets (SIPs) come with AmigoSH to better support the MOBILEROBOTS community. On request from the client by a related AmigoSH command, the AmigoSH server packages and sends one or a continuous stream of information packets to the client over the HOST serial communication line. Extended packets get sent immediately before (such as `GYROPac`) or after (such as `IOPac`) the standard SIP that AmigoSH sends to your client every SIP milliseconds.¹³

The standard SIP takes priority so you may have to adjust the HOST serial baud rate to accommodate all data packets in the allotted cycle time, or some packets may never get sent.

Packet Processing

Identical with the standard SIP, all AmigoSH server information packets get encapsulated with the header (0xFA, 0xFB; 250, 251), byte count, packet type byte and trailing checksum. It is up to the client to parse the packets, sorted by type for content. Please consult the respective client application programming manuals for details.

CONFIGpac and CONFIG Command

Send the `CONFIG` command #18 without an argument to have AmigoSH send back a `CONFIGpac` SIP packet type 32 (0x20) containing the robot's operational parameters. Use the `CONFIGpac` to examine

¹³ You may have to adjust the HOST serial baud rate to accommodate the additional communications traffic.

many of your robot's default FLASH_based settings and their working values, where appropriate, as changed by other client commands, such as SETV and ROTKV.

Table 7. CONFIGpac contents

LABEL	DATA	DESCRIPTION
Header	int	Common packet header = 0xFAFB
Byte count	byte	Number of following data bytes
Type	byte	ENCODERpac = 0x20
Robot type	str	Typically "Pioneer"
Subtype	str	Identifies the ActivMedia robot model; e.g. "AmigoBot-SH",
Sernum	str	Serial number for the robot.
4mots	byte	Antiquated (=1 if AT with P2OS)
Rotveltop	int	Maximum rotation velocity; deg/sec
Transveltop	int	Maximum translation speed; mm/sec
Rotacctop	int	Maximum rotation (de)acceleration; deg/sec ²
Transacctop	int	Maximum translation (de)acceleration; mm/sec ²
PWMmax	int	Maximum motor PWM (limit is 500).
Name	str	Unique name given to your robot.
SIPcycle	byte	Server information packet cycle time; ms.
Hostbaud	byte	Baud rate for client-server HOST serial: 0=9.6k, 1=19.2k, 2=38.4k, 3=56.8k, 4=115.2k.
Auxbaud	byte	Baud rate for AUX1 serial port; see HostBaud.
Gripper	int	0 if no Gripper; else 1.
Front sonar	int	1 if robot has front sonar array enabled, else 0.
Rear sonar	byte	1 if robot has rear sonar enabled, else 0.
Lowbattery	int	In 1/10 volts; alarm activated when battery charge falls below this value.
Revcount	int	Working number of differential encoder ticks for a 180 degree revolution of the robot.
Watchdog	int	Ms time before robot automatically stops if it has not received a command from the client. Restarts on restoration of connection.
P2mpacs	byte	1 means alternative SIP enable; not used by AmigoSH.
Stallval	int	Maximum PWM before stall. If > PWMMAX, never.
Stallcount	int	Ms time after a stall for recovery. Motors lax during this time.
Joyvel	int	Joystick translation velocity setting, mm/sec
Joyrvel	int	Joystick rotation velocity setting in deg/sec
Rotvelmax	int	Current max rotation speed; deg/sec.
Transvelmax	int	Current max translation speed; mm/sec.
Rotacc	int	Current rotation acceleration; deg/ sec ²
Rotdecel	int	Current rotation deceleration; deg/ sec ²
Rotkp	int	Current Proportional PID for rotation
Rotkv	int	Current Derivative PID for rotation
Rotki	int	Current Integral PID for rotation
Transacc	int	Current translation acceleration; mm/ sec ²
Transdecel	int	Current translation deceleration; mm/ sec ²
Transkp	int	Current Proportional PID for translation.
Transkv	int	Current Derivative PID for translation.
Transki	int	Current Integral PID for translation.
Frontbumps	byte	Number of front bumper segments.
Rearbumps	byte	Number of rear bumper segments.
Charger	byte	1 if P3/PeopleBot or 2 if PowerBot automated charger mechanism and circuitry installed in robot; otherwise 0.
Sonarcycle	byte	Sonar duty cycle time in milliseconds.
Autobaud	byte	1 if the client can change baud rates; 2 if auto-baud implemented.
Hasgyro	byte	1 if robot equipped with the gyro heading correction device; otherwise 0.
Driftfactor	int	Working drift factor value.
Aux2baud	byte	Baud rate for AUX2 serial port; see HostBaud.
Aux3baud	byte	Baud rate for AUX3 serial port; see HostBaud.
Ticksmm	int	Encoder ticks per millimeter tire motion
Shutdownvolts	int	DC volts X10 at or below which the onboard PC will shut down
WITH VERSION 1.5...		
Versionmajor	str	Null-terminated strings for AmigoSH version numbers
Versionminor	str	
Chargethreshold	int	DC volts X10 for PowerBot's charger threshold.

SERIAL

The baud rates for the HOST and AUX serial ports initially are set from their respective FLASH-based defaults and get reset to those values whenever the microcontroller is reset or upon client disconnection. For advanced serial port management from the client side, AmigoSH provides four client commands which let your software reset the HOST (`HOSTBAUD #50`), Aux1 (`AUX1BAUD #51`), Aux2 (`AUX2BAUD #52`) and Aux3 (`AUX3BAUD #53`) serial port baud rates, respectively. Use the integer command argument values: 0=9600, 1=19.2K, 2=38.8K, 3=57.6K, or 4=115.2K baud, respectively.

With auto-bauding, the HOST serial port automatically reverts to its FLASH default baud rate if, after being reset by the `HOSTBAUD` client command, it does not receive a subsequent and valid client-command packet within 500 milliseconds.

HOST-to-AUX Serial Transfers

Use the client-side `TTY2` command #42 with a string argument to have that string sent out the Aux1 port to the attached serial device, such as a robotic camera. AmigoSH also maintains a circular buffer for incoming serial data from the respective Aux1 port. On request, AmigoSH sends successive portions of the buffer to your client via the HOST serial in the respective `SERAUXpac` (type = 176; 0xB0) SIP.

Use the integer argument value of zero to flush the contents of the buffer. Otherwise, use an argument value of up to 253 bytes to have AmigoSH wait to collect the requested number of incoming AUX1-port serial bytes and then send them in the respective `SERAUXpac` SIP.

ENCODERS

Issue the `ENCODER` command #19 with an argument of one for a single or with an argument value of two or more for a continuous stream of `ENCODERpac` (type 144; 0x90) SIPs. Discontinue the packets with the `ENCODER` command #19 with an argument of zero.

Table 8. *ENCODERpac* SIP contents

LABEL	DATA	DESCRIPTION
Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes + 2 (checksum)
Type	byte	0x90
Left Encoder	integer	Least significant, most significant portion of the current accumulated encoder counts from the left wheel
	integer	
Right Encoder	integer	Least significant, most significant portion of the current accumulated encoder counts from the right wheel
	integer	
Checksum	integer	Checksum for packet integrity

BUZZER SOUNDS

AmigoBots have a piezo buzzer on the microcontroller that aurally notifies you of system conditions, such as low batteries or stalls. For stealthy operation, issue the `SOUNTOG` command #92 with an argument of zero to mute the microcontroller's buzzer or argument of one to re-enable it. (See also the `SOUNTOG` FLASH parameter in the next chapter to set its default state.)

The `SAY` command #15 lets you play your own sounds through the buzzer. The argument consists of a length-specified string of duration and tone pair bytes. The duration is measured in 20 millisecond increments.

A tone value of zero means silence (musical rest). The next 127 frequencies (1-127) are the corresponding MIDI notes. The remaining tones are frequencies computed as:

$$\text{Tone} - 127 * 32$$

equivalent frequencies from 1 to 4096, in 32 Hz increments.

Except for the MIDI notes, you'll just have to experiment with tones. Here is the sequence that generates the AmigoSH distress wail when the robot stalls or the batteries are low:

```
50,100,20,0,50,60,0
```

TCM2

The TCM2 accessory is an integrated inclinometer, magnetometer, thermometer and compass that attaches to the Aux1 serial port of the AmigoSH microcontroller. When attached and enabled, special TCM2 compass servers read and report the heading in ± 2 degree increments as the `compass` byte in the standard SIP. Use the `TCM2` command 45 to request additional information from the device in the form of the `TCM2pac`. See the *TCM2 Manual* and supporting software that accompanies the device for details.

Heading Correction Gyro

With the gyroscope accessory, your client software may detect and compensate for robot heading changes that aren't detected by the wheel encoders, such as from slipping wheels. The microcontroller supports the gyro via attachment to its AN6 and AN7 analog-to-digital input ports.

AmigoSH collects 10-bit (0-1023) gyro rate and 8-bit (0-255) temperature data and will, upon request, send the collected data to a connected client in a `GYROPac` (type=0x98) server information packet for processing. Analysis of the gyro data and subsequent modifications to the robot's heading are done on the client side, as supported in the latest versions (1.3 and later) of ARIA.

To enable the gyro, you must set the `hasGyro` FLASH parameter to one using the AmigoSHcf tool (see next chapter). Set it to zero if the gyro isn't attached. Then to acquire gyro data, send the `GYRO` command #58 with integer argument of one; zero disables the gyro SIP. The gyro SIP is stopped upon client disconnection or microcontroller reset, too.

AmigoSH collects the gyro rate and temperature readings at the maximum rate of once every 25 milliseconds and reports each of these values to the client, when enabled, in the `GYROPac` SIP that gets sent just before the standard Server Information Packet every `sInfoCycle`, typically every 100ms. `GYROPac` consists of a count byte of the rate and temperature data pairs accumulated since the last cycle (typically four for a 100ms cycle time), followed by that number of rate/temperature integer/byte pairs.

Gyro rates are 10-bit integers of value 0-1023. When not moving, the rate is centered around 512 or so, depending on the gyro's temperature and other calibration factors which drift with use and should be corrected on the fly. Values below that center point indicate counter-clockwise rotation rates; values above the resting center measure clockwise rotation rates.

Table 9. *GYROPac SIP contents*

LABEL	DATA	VALUE	DESCRIPTION
Header	2	0xFA, 0xFB	Common header
Byte count	1	xx	Varies
Type	1	0x98	Packet type
N pairs	1	x	Number of gyro data pairs
FOR N PAIRS			
Rate	2	varies 0-1023	Gyro rate
Temperature	1	varies 0-255	Gyro temperature
Checksum	2	varies	Computed checksum

Chapter 6 Updating & Reconfiguring AmigoSH

The AmigoSH firmware and a set of operating parameters get stored in your Team AmigoBot microcontroller's FLASH. With special upload and configuration software tools, you change and update AmigoSH and its operating parameters. No hardware modification is required.

WHERE TO GET AMIGO^{SH} SOFTWARE

Your AmigoBot comes installed with the latest version of AmigoSH. And the various AmigoSH configuration and update tools come with the robot on CD-ROM. Thereafter, stay tuned to the `amigobot-users` newsgroup or periodically visit our support website to obtain the latest AmigoSH software and related documentation:

`http://robots.MobileRobots.com`

The main utility, `AmigoSHcf`, is a multi-functional application for both uploading new AmigoSH versions as well as modifying your robot's onboard FLASH-based parameters.

AMIGO^{SH} MAINTENANCE MODE

To connect with and update your robot's AmigoSH servers and its FLASH-based operating parameters, you need to first connect a serial port on the PC from which you will run `AmigoSHcf` to the `Host` port of your AmigoBot just beneath the access portal. Unplug the Ethernet-to-serial device, if you have one installed.

Enabling Maintenance Mode on the Controller

You have three ways in which to put the controller into AmigoSH Maintenance Mode:

- ✓ Start Up
- ✓ Manual
- ✓ Automatic

If for any reason your robot's FLASH parameters get erased or your AmigoSH software encounters a code fault, your AmigoBot controller automatically reverts to AMIGOstub-based maintenance mode.¹⁴ Or if you attach a PC to the `Host` port in the portal, open that port through software such as by running `AmigoSHcf` on that PC, and then reset or otherwise restart the controller, it will automatically revert into maintenance mode.

Like with previous AmigoBot and Pioneer controllers, you also may engage maintenance mode manually:

1. Press and hold the `MOTORS/TEST` button on the User Control Panel
2. Press and release the adjacent `RESET` button
3. Release the `MOTORS/TEST` button.

Unlike previous controllers, and certainly with much more convenience since you don't need to be right next to the robot, `AmigoSHcf` *automatically* engages maintenance mode when run through an onboard PC. Just start it up and it forces the controller into maintenance mode. (It uses the AmigoSH client command `#255` to do that.)

AMIGO^{SH}CF

The AmigoSH update and configuration program, `AmigoSHcf`, is part of a collection of utilities and files for comprehensive management of your robot's onboard servers and FLASH-based operating parameters. The distribution archive for the software is simply named `AmigoSHV_v` (`v` and `v` are the version major and minor numbers, such as `1_0`), with a `.tgz` suffix for Linux-based PCs or `.exe` for Windows computers.

¹⁴ AMIGOstub is a resident GDB-like interface for FLASH updates and other debugging uses.

Install the utilities and files on the PC you plan to use for maintaining your robot's operating system and parameters by double-clicking the distribution software's onscreen icon or otherwise executing the self-extracting, self-installing package. For Linux, `uncompress` and `untar` the files. For example,

```
% tar -zxvf AmigoSH1_0.tgz
```

The expanded archive creates an `AmigoSH/` directory in the selected Windows or current Linux path and stores the AmigoSH software within.

STARTING AMIGOSHcf

`AmigoSHcf` is a text-based console application which, like `demo`, is built with ARIA. It runs in two stages: Startup followed by interactive mode. When invoked, you may start `AmigoSHcf` with the various ARIA command-line options. With an X-terminal under Linux, for example, navigate to the `AmigoSH` directory and invoke the program:

```
% cd /usr/local/AmigoSH
% ./AmigoSHcf <options>
```

With Windows PCs, you may double-click the `AmigoSHcf` icon to automatically open a console window and start the program without any options. To start up with command-line options, Run the program from the `Start` menu, or run `Command` from the `Start` menu, then navigate to the `AmigoSH` directory and start `AmigoSHcf` with options.

For example (after invoking the MSDOS-like command window):

```
C:\> cd Program Files\MobileRobots\AmigoSH
C:\Program Files\MobileRobots\AmigoSH> AmigoSHcf <options>
```

Normally (without any command-line arguments), `AmigoSHcf` starts up expecting to connect your PC's COM1 or `/dev/ttyS0` serial port with your robot's controller. If successfully connected, the program automatically retrieves your robot's FLASH-stored operating parameters and enters interactive mode.

Table 10. *AmigoSHcf startup options*

KEY	ARGUMENT	DESCRIPTION
-b	command arguments	Batch mode executes list of <code>AmigoSHcf</code> interactive-like mode commands with arguments, then exits automatically.
-u	motfile	Automatically upload <code>AmigoSH (.mot)</code> motfile after connecting with the controller.
-l	paramsfile	Load the disk-stored params <code>(.rop)</code> file instead of the robot's copy
-n	none	Don't automatically connect with controller
-rp	serial port	Uses specified serial port for connection
-rb	baud rate	Specify the port connection baud rate
-s	paramsfile	On exit from <code>AmigoSHcf</code> , automatically save the current parameter values to the named <code>.rop</code> paramsfile

For instance, operate `AmigoSHcf`'s interactive features without a connection to maintain disk-based copies of your robot's operating parameters.

Include each of the selected `AmigoSHcf`'s startup-mode options as a key letter with a dash ("-") prefix, followed by any required arguments, separated by spaces. For example, to start up `AmigoSHcf` and make a connection with a serial port other than the default COM1 or `ttyS0`:

```
C:\Program Files\MobileRobots\AmigoSH> AmigoSHcf -p COM3
```

Similarly, in Linux, the command uploads a fresh copy of `AmigoSH` to your robot's controller and then exits automatically:

```
% ./AmigoSHcf -u AmigoSH1_0.mot -n -b
```

CONFIGURING AMIGO SH PARAMETERS

Your `AmigoBot` has several parameters stored in FLASH that `AmigoSH` uses to configure its servers and auxiliary attachments and to uniquely identify your robot. For instance, the default maximum translation velocity is stored in the `TransVelMax` parameter. Its value takes effect when starting your robot or after resetting the controller, and may be changed temporarily by a client command. Use

Updating and Reconfiguring AmigoSH

AmigoSHcf's batch or interactive modes to modify these operating parameters, and hence your robot's default operating characteristics.

Start up AmigoSHcf as described in the previous section. As discussed earlier, AmigoSHcf normally downloads the set of operating parameters from your robot's FLASH for your review and modification. Or you may load a disk-stored version of those parameters.

Interactive Commands

To operate AmigoSHcf in interactive mode, simply type a keyword at the command line. Some keywords affect the operation of AmigoSHcf, the status of the parameters file as a whole, or the connection between AmigoSHcf and your robot's controller. For instance, to review the list of current AmigoSH FLASH variables, type 'v' or 'view' followed by a return (Enter). Each successive return will display additional variable keywords and current values. Similarly, type '?' or 'help' to see a list of AmigoSHcf interactive commands.

Changing Parameters

Most keywords refer to the operating parameters themselves. Alone, a parameter's keyword simply asks AmigoSHcf to display the parameter's value. Provide an argument with the parameter keyword separated by a space to change its value. That value may be a string (no quotes or spaces) or a decimal or hexadecimal ("0xN") number. For example, to change the watchdog timeout to four seconds, type:

```
> watchdog 4000
```

or

```
> watchdog 0xfa0
```

See the respective control command and parameter Tables nearby for a full description of AmigoSHcf operation.

Table 11. AmigoSHcf control commands

COMMAND	DESCRIPTION
keyword <value>	Alone, a keyword displays current, edited value. Add argument to change current value.
v or view	Display FLASH parameters; use ENTER to continue display list.
u or upload <motfile>	Upload specified AmigoSH image (.mot) file to the controller.
r or restore <paramsfile>	Restore parameters to values currently stored in FLASH or, if given, from a paramsfile (.rop) on disk
save <paramsfile>	Saves current edited values to FLASH or saves current edited values to paramsfile (.rop) on disk for later reference.
q or quit	Exits AmigoSHcf.
? or help	Displays these commands and descriptions.

SAVE YOUR WORK

While changing parameter values in AmigoSHcf interactive mode, you are editing a temporary copy; your changes are not put into effect in your robot's FLASH until you explicitly "save" them to the controller.

Also use the AmigoSHcf save command to save a copy of the parameters to a disk file for later upload. We strongly recommend that you save each version of your robot's parameter values to disk for later retrieval should your controller get damaged or its FLASH inadvertently erased. Default

parameter files come with each AmigoSH distribution, but it is tedious to reconstruct an individual robot's unique configuration.

Table 12. AmigoSH FLASH configuration parameters

KEYWORD	TYPE	DEFAULT	DESCRIPTION
type	str	Pioneer	Identifies the robot type.
subtype	str	Amigo-sh	Identifies the robot model.
name	str	not set	Unique name for your robot. Maximum of 20 characters, no spaces.
Sernum	str	factory	Serial number for the robot.
ticksmm	int	123	Encoder ticks/mm: (ticks per rev x gear-ratio) (wheel diameter x PI)
revcount	int	11548	The number of differential encoder ticks for a 180 degree revolution of the robot.
driftfactor	int	0	Value in 1/8192 increments to be added or subtracted from the left encoder ticks in order to compensate for tire differences.
lowbattery	int	117	In 1/10 volts; controller alarm activated when battery charge falls below this value.
shutdownvolts	int	110	In 1/10 volts; controller disconnects client when battery charge falls below this value.
hostbaud	byte	0	Baud rate for client-server HOST serial: 0=9.6k, 1=19.2k, 2=38.4k, 3=56.8k, 4=115.2k.
auxbaud1	byte	0	Baud rate for AUX serial port 1; see HostBaud
sipcycle	byte	100	Server information packet cycle time in 1 ms increments. Default is classic 100 ms.
watchdog	int	2000	Ms time before robot automatically stops if it has not received a command from a client. Restarts on restoration of connection.
soundtog	byte	1	0 disables the buzzer
sonarcycle	byte	40	Sonar cycle time in milliseconds
sonar1	str	12345678	Ping sequence for sonar array #1. Up to 16 number characters 1-8; 0 to disable the array
hasgyro	byte	0	Set to 1 if you have the gyro accessory
tcm2	byte	0	TCM2 module, if connected, specify AUX port 1,2, or 3
lcd	byte	0	Aux port number of attached LCD module
stallval	int	400	Maximum PWM before stall. If > PwmMax, never.
stallcount	int	200	Ms time after a stall for recovery. Motors not engaged during this time.
pwmmax	int	500	Maximum motor PWM (500 maximum).
rotveltop	int	360	Maximum rotation velocity; deg/sec
transveltop	int	1074	Maximum translation speed; mm/sec
rotacctop	int	324	Maximum rotation (de)acceleration; deg/sec ²
transacctop	int	1000	Maximum translation (de)acceleration; mm/sec ²
rotvelmax	int	324	Max rotation speed; deg/sec.
transvelmax	int	1000	Max translation speed; mm/sec.
rotacc	int	162	Rotation acceleration; deg/sec ²
rotdecel	int	162	Rotation deceleration; deg/sec ²
rotkp	int	23	Proportional PID for rotation
rotkv	int	3	Differential PID for rotation
rotki	int	0	Integral PID for rotation
transacc	int	500	Translation acceleration; mm/sec ²
transdecel	int	500	Translation deceleration; mm/sec ²
transkp	int	42	Proportional PID for translation
transkv	int	3	Differential PID for translation
transki	int	0	Integral PID for translation

PID PARAMETERS

The AmigoSH configuration parameters include settings for the PID motor controls for translation and rotation of the robot. The translation values also are used for independent-wheel mode. The default values are for a lightly loaded robot. Experiment with different values to improve the performance of your robot in its current environment.

The Proportional PID (Kp) values control the responsiveness of your robot. Lower values make for a slower system; higher values make the robot "zipper", but can lead to overshoot and oscillation.

The Derivative PID (Kv) dampens oscillation and overshoot. Increasing values gives better control of oscillation and overshoot, but they also make the robot's movements more sluggish.

Updating and Reconfiguring AmigoSH

The Integral PID (Ki) adjusts residual error in turning and velocity. Higher values make the robot correct increasingly smaller errors between its desired and actual angular position and speed.

DRIFTFACTOR, TICKSMM AND REVCOUNT

AmigoSH uses the `ticksMM` and `revCount` parameters to convert your platform-independent speed and rotation commands—typically expressed in millimeters or degrees, respectively—into platform-dependent units. And it uses the `driftFactor` to compensate for tire difference.

The `ticksMM` value is the number of encoder pulses (“ticks”) per millimeter of wheel rotation. The value is, of course, dependent upon the wheel encoder’s resolution, the motor-to-wheel gear ratio, and the wheel’s diameter.

The `revCount` value is the number of encoder ticks for one-half revolution of the robot. It depends on a number of factors, principally the length of the wheel base, which may change due to payload, tire wear, operating surface, and so on.

The `driftFactor` is a value in 1/8192 units that gets added or subtracted from the left-wheel encoder count at each motor cycle. In doing so, it compensates for tire difference and thereby straightens the robot’s translation forward and backward.

The `ticksMM` and `revCount` parameters affect the conversion of your motion command arguments into platform-dependent values used by AmigoSH. Unlike previous controller software, AmigoSH also uses `ticksMM` and `revCount` to convert its internal measures into platform-independent position, heading, and velocity values reported back from the server, such as `X-Pos` and `Th`. Accordingly, you’ll notice that the respective ARIA client parameters have many conversion factors like `DistConvFactor` set to 1.0.

STALLVAL AND STALLCOUNT

An AmigoSH stall monitor maintains a running average of PWM values for each wheel over a 500 millisecond integration period. PWM values get added to the sum if the wheel speed is below 100 mm/sec. The average is then compared with the `stallval` FLASH value. If it exceeds that value, in other words the motors are being given lots of power but are barely moving if at all, a stall occurs. Once stalled, power is removed and the motors relax for the `stallwait` period, after which power gets reapplied.

Chapter 7 Calibration & Maintenance

Your *Team AmigoBot* is built to last a lifetime and requires little maintenance.

CALIBRATING YOUR ROBOT

Your robot comes with FLASH parameters adjusted for operation on smooth, flat surfaces with its original payload. If you operate your robot on some different surface and with lighter or heavier loads, you probably will need to recalibrate many of its operating parameters, such as `driftFactor`, `revCount`, `ticksMM` and the PIDs.

The ARIA `demo` program has two modes to help you do that. In 'p'osition mode, `demo` displays current heading and position. Press the 'r' key to reset these to 0 at any time. Press the right or left arrow key to have the robot rotate 90 degrees either clockwise or counterclockwise, respectively. The up and down arrow keys tell the robot to advance forward or backward one meter, respectively.

ARIA `demo`'s 'd'irect mode lets you change the variety of operating parameters on-the-fly by sending the respective AmigoSH client command number and value to be used during the current session. To replace the default values for permanent use by AmigoSH, use AmigoSHcf.

Accordingly, to properly calibrate your robot, first use AmigoSHcf and record the default values for the respective parameters, such as for `driftFactor` and `revCount`. Then connect with the ARIA `demo` in position mode and move the robot. As accurately as possible, measure its actual motion and position and use direct mode to adjust the reported values for your robot's current configuration and operating environment.

For example, start with `driftFactor` since its value affects both `ticksMM` and `revCount`. Draw a line on the floor parallel to the robot's translation travel and drive the robot forward or back at least five meters. Adjust `driftFactor` (command #89) to minimize the robot's drift off that line.

Then drive the robot forward or back one or more meters and compare its actual translation distance you accurately measure with `demo`'s AmigoSH-reported distance (x) in millimeters. Adjust `ticksMM` (command #93) so that the numbers match.

Likewise, rotate your robot and compare your measured rotation angle to the reported heading (`th`). Adjust `revCount` (the measure of differential encoder ticks to achieve 180-degrees rotation) accordingly (command #88).

Finally, drive the robot around and adjust its PID, velocity and acceleration values to achieve the desired performance for the operating configuration.

When you are satisfied that the robot moves and rotates the proper distances and headings, and drives with the proper performance, commit the new values into their related FLASH parameters in your robot with AmigoSHcf and don't forget to save a copy in a `.rop` file for later reference.

DRIVE LUBRICATION

The drive motors and gearbox are sealed and self-lubricating, so you need not fuss with grease or oil. An occasional drop or two of oil on the axle bushings between the wheels and the case won't hurt.

Keep the axles clear of carpet or other strings that may wrap around and bind up AmigoBot's drive. Occasionally wipe the tires with a damp cloth, and especially remove any dirt or debris that may accumulate on the tires—these will degrade the robot's performance.

AMIGOBOT BATTERIES

Lead-acid batteries like those in your car and in your AmigoBot, last longest when kept fully charged. In fact, severe discharge is harmful to the battery. So be careful not to operate the robot if the battery voltage falls below 11 volts or so. (The robot is programmed to stop working at 10 volts). In other words, heed your robot's incessant whining when its batteries are weak.

**Severe discharge is harmful to the battery.
Store the battery fully charged.**

It's also a good idea to store the robot plugged into the charger when it's not going to be used for a day or two. Charge the battery fully, then store the robot in a cool, dry place if you intend not to use the robot for any longer period of time(a pity!), such as for a week or more.

Charging the Battery

If you have the standard AmigoBot or the high-speed charger accessory, insert it into a common 120-volt (European 240) AC power socket. (Some users may require a special power-socket adapter which accompanies the charger.) Then insert the charger's cable into the Charge socket that is next to AmigoBot's Power switch on the underside of the robot. With the high-speed charger accessory, its LEDs indicate charge status, as marked on its case.

It takes fewer than 12 hours—often just a few hours, depending on the level of discharge— to fully charge the AmigoBot battery using its standard charger (roughly, three hours per volt). The fast-charger accessory can charge the battery in just an hour or two.

You do not have to turn off your robot when it is charging. In fact, we encourage you to plug in the charger while you are programming the robot. Just make sure to disable the robot's motors. For instance, put it up on blocks if you put store the robot on a table for charging. And realize that the charging time will roughly double if you continue to operate your robot while charging.

Alternative Battery Chargers

The chargers that we supply for the AmigoBot are safe: They pose no danger to the user, they properly charge the robot's battery, and they may be left on and connected with the robot for many days.

**Use a different battery charger
AT YOUR OWN RISK.**

The center post of the charger socket on the AmigoBot is the positive (+) side of the battery; the shaft is the negative (-) side. If you choose to use an alternative battery charger for AmigoBot, be sure to connect positive to positive and negative to negative from charger to AmigoBot.

An alternative AC to DC converter/battery charger for AmigoBot should sustain at least 0.2 A at 13.75 to 14 VDC per battery. It also should be voltage- and current-limited so that it cannot overcharge the batteries.

GETTING INSIDE

We discourage you from opening up your AmigoBot. Period.

**Open the robot
AT YOUR OWN RISK,
unless explicitly authorized by MOBILEROBOTS support.**

FACTORY REPAIRS

If after reading this manual, you're having *hardware* problems with your AmigoBot and you're sure that it needs repair, contact us:

<http://robots.activmedia.com/techsupport>

In the body of your e-mail message, provide your robot's serial number found on its underside, and describe the problem you are having in as much detail as possible. Also include your name, e-mail

and mail addresses, as well as phone and fax numbers. Tell us when and how we can best contact you. We will assume e-mail is the best manner, unless otherwise notified.

We will try to resolve the problem through communication. If the robot must be returned to the factory for repair, obtain a shipping and repair authorization code and shipping details from us first.

ATTENTION!
Obtain a Repair or Replacement Authorization Code
before returning any defective items.

MOBILEROBOTS is not responsible for shipping damage or loss, we will assist you in tracking your robot through the shipper.

Appendix A

CONTROLLER PORTS & CONNECTIONS

This Appendix contains pinout and electrical specifications for the external and internal ports and connectors on the SH2-based controller for AmigoBot.

Note that layered connectors are numbered differently, depending on the socket type. IDC ones are odd and even layers; microfit connectors use successive-position numbering. See the Figures nearby for examples.

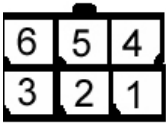


Figure 18. Mini- and micro-fit style connector numbering

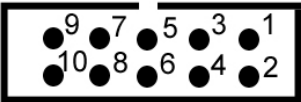


Figure 19. IDC-type connector numbering

AMIGOSH CONTROLLER

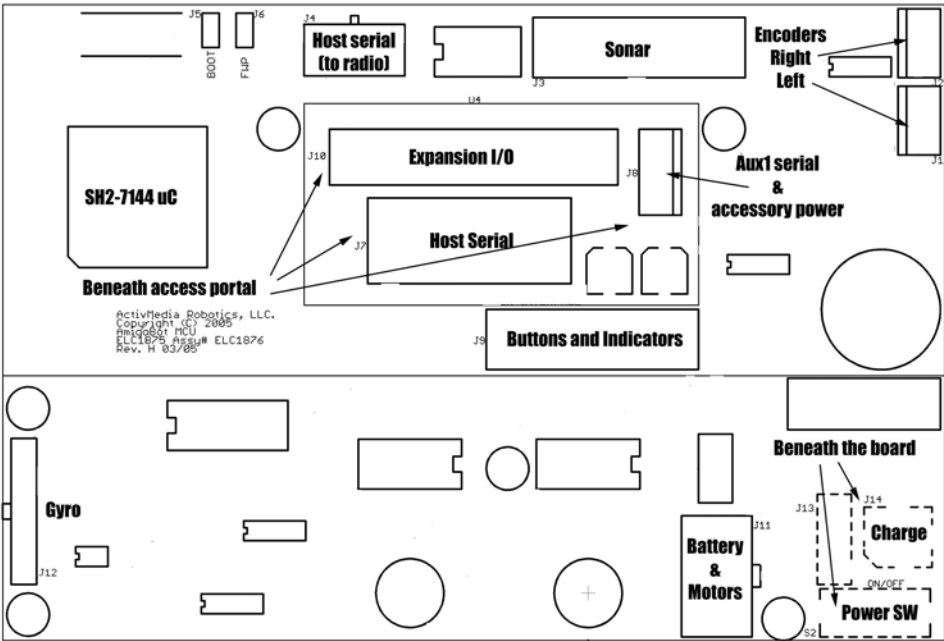


Figure 20. Key components and connectors on AmigoBot's SH2-based controller.

Battery and Motors Connector

The battery and motors connector is an 8-position minifit socket for power connections to the battery, controller, and motors.

Table 13. Battery and Motors Connector (rev H)

PIN	SIGNAL	DESCRIPTION	PIN	SIGNAL	DESCRIPTION
1	LMR	Left motor red	2	LMB	Left motor black
3	RMR	Right motor red	4	RMB	Right motor black
5	Common	Charger +	6	VBatt	Battery +
7	n/c		8	Charger	Charger +

Serial Ports

The SH2 supports four SCI serial ports. The AmigoBot-SH board implements two of them for RS232 connections. The HOST port (SCI1) for client-server and maintenance communications is shared between two connectors:¹⁵ a 9-position female DSUB accessible through AmigoBot's access portal and an 8-position microfit which cable, when installed, delivers power and signals to a device underneath and to the front of AmigoBot; typically to the Ethernet-to-serial radio accessory.

Another 5-position microfit socket accessible through the portal on top of AmigoBot provides an Auxiliary 1 (SCIO) RS232 serial signals and power for external accessories, such as an LCD or TCM2 module. Raw SCI transmit and receive signals for two more serial ports, Aux2 and Aux3, are found on the Expansion I/O connector (table below).

The HOST serial connector through the access portal also has a DSR signal line that is set to RS232 high (+8 - 15VDC) when the controller is powered on. The HOST serial connectors are wired DCE for direct connection (straight-through cable, not NULL-modem) to a standard PC serial port.

Table 14. HOST serial port through AmigoBot's access portal (female DSUB-9 socket; J7)

PIN	SIGNAL	DESCRIPTION	PIN	SIGNAL	DESCRIPTION
1	nc		2	TXD	RS232 transmit signal out
3	RCV	RS232 receive signal in	4	nc	
5	GND	Signal power common	6	DSR	RS232 high when controller powered
7	nc		8	nc	
9	nc				

† Shared with internal HOST serial

Table 15. Internal HOST serial (8-pos microfit; J4)

PIN	SIGNAL	DESCRIPTION	PIN	SIGNAL	DESCRIPTION
1	GND	Common	2	TXD	RS232 transmit signal out
3	RCV	RS232 receive signal in	4	DTR	Signal in*
5	Vcc	5VDC @ 1A	6	GND	Common
7	Vbatt	12VDC @1A	8	DSR	RS232 high when controller powered

*When active high, switches this HOST's TXD and RCV signals in to µC

Table 16. Aux1 serial port (5-pos microfit sockets)

PIN	SIGNAL	DESCRIPTION	PIN	SIGNAL	DESCRIPTION
1	Vcc	5 VDC	2	TXD	Signal out
3	RCV	Signal in	4	Vbatt	12 VDC
5	GND	Common			

Encoders

Table 17. Encoder connectors (2 x 4-pos microfits)

PIN	SIGNAL	DESCRIPTION	PIN	SIGNAL	DESCRIPTION
1	Vcc	5VDC	2	CHA	Channel A
3	GND	Common	4	CHB	Channel B

Sonar

Table 18. Sonar connector (10-pos latching IDC)

PIN	SIGNAL	DESCRIPTION	PIN	SIGNAL	DESCRIPTION
1	A0	disc address	2	A1	disc address
3	A2	disc address	4	BINH	inhibits return signal
5	INIT	starts sonar ping	6	VCC	5 VDC
7	VCC	5 VDC	8	SGND	Common
9	SGND	Common	10	ECHO	Goes high when echo threshold reached

¹⁵ A high signal on the internal HOST port's DTR pin 4 electronically switches its TXD and RCV signals in to the controller. Otherwise, the signals are isolated from the shared HOST port accessed through the top portal.

Buttons and Indicators

Table 19. User control buttons and indicators

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	SYSLED	System LED		2	ULED	User LED
3	Vp	3.3VDC		4	/MTPB	Motors/Test pushbutton
5	/RST	Reset pushbutton		6	Vp	3.3VDC
7	nc			8	GND	Common
9	P1	Piezo @ VBATT+		10	P2	Piezo -

Heading Correction Gyro

Table 20. Heading correction gyro connector (6-pos microfit)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	nc			2	VCC	5 VDC power
3	RATE	AN6		4	TEMP	AN7
5	AGND	Analog gnd		6	GND	Power ground

Expansion I/O

Table 21. Expansion I/O (40-pos high-density latching IDC)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	Vbatt	12VDC		2	GND	Common
3	Vbatt	12VDC		4	GND	Common
5	Vbatt	12VDC		6	GND	Common
7	Vcc	5VDC		8	Vp	3.3VDC
9	Vcc	5VDC		10	Vp	3.3VDC
11	GND	Common		12	AGND	Analog common
13	nc			14	AN2	Analog 2
15	GND	Common		16	An1	Analog 1
17	D7	Data 7		18	AN0	Analog 0
19	D6	Data 6		20	AGND	Analog common
21	D5	Data 5		22	SDA	I2C data
23	D4	Data 4		24	SCL	I2C clock
25	D3	Data 3		26	TXD3	Aux3 serial transmit
27	D2	Data 2		28	RXD3	Aux3 serial receive
29	D1	Data 1		30	TXD2	Aux2 serial transmit
31	D0	Data 0		32	RCV2	Aux2 serial receive
33	/CS1	Chip select 1		34	/WRL	Write
35	/CS0	Chip select 0		36	/RD	Read
37	A3	Address 3		38	A0	Address 0
39	A2	Address 2		40	A1	Address 1

Appendix B

SPECIFICATIONS

Physical Characteristics

Length	33 cm
Width	28 cm
Height (body)	13 cm
Body clearance	3 cm
Weight	3.6 Kg
Payload	1 Kg

Construction

Body	Molded polycarbonate
Chassis	1.6mm CNC fabricated aluminum
Assembly	Allen hex screws (metric)

Power

Battery	2x12V lead-acid
Charge	55.2 watt-hr
Run time	3+ hours
Recharge time (trickle)	12 hrs
Recharge time (fast)	4 hrs

Mobility

Drive wheels	2 solid rubber, with caster balance
Wheel diameter	10.2 cm
Wheel width	3 cm
Steering	Differential
Gear ratio	19.5:1
Swing radius	33 cm
Turn radius	0 cm
Translate speed max	1000 mm/sec
Rotational speed max	100 degrees/sec
Traversable step max	1.5 cm
Traversable terrain	All wheelchair accessible

Sensors

Sonar	8 total 1 each side 4 forward 2 rear
Position encoders	2 (one each motor) 39,000 ticks per wheel revolution 124 ticks per mm

Electronics

Processor	44 MHz Renesas SH2-7144
Position inputs	4
Data bus	8-bit R/W 2 x chip selects
Analog-to-digital	3 x 1024-bit with isolated analog-ground
Sonar	1 x 8 (multiplexed)
Comm port	2 x RS-232 2 x SCI

FLASH	128 KB
RAM	32 KB

Controls and Indicators

Power	Robot/accessories power ON/OFF
Charge	System power/battery recharge
Reset	Warm reboot/download
Motors/Test	Motors/maintenance/self-tests
Speaker	Piezo

Warranty & Liabilities

Your MobileRobots platform is fully warranted against defective parts or assembly for one year after it is shipped to you from the factory. Accessories are warranted for 90 days. Use only MobileRobots authorized parts or warranty void. This warranty also explicitly *does not include* damage from shipping or from abuse or inappropriate operation, such as if the robot is allowed to tumble or fall off a ledge, or if it is overloaded with heavy objects.

The developers, marketers and manufacturers of MobileRobots products shall bear no liabilities for operation and use of the robot or any accompanying software except that covered by the warranty and period. The developers, marketers or manufacturers shall not be held responsible for any injury to persons or property involving MobileRobots products in any way. They shall bear no responsibilities or liabilities for any operation or application of the robot, or for support of any of those activities. And under no circumstances will the developers, marketers or manufacturers of MobileRobots products take responsibility for support of any special or custom modification to MobileRobots platforms or their software.



19 Columbia Drive
Amherst, NH 03031
(603) 881-7960
(603) 881-3818 fax
<http://www.MobileRobots.com>