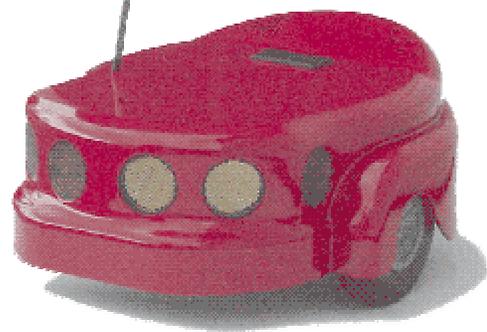


***Amigo*BOT**

Technical Manual



Copyright 2002, ActivMEDIA ROBOTICS, LLC. All rights reserved.

Under international copyright laws, this manual or any portion of it may not be copied or in any way duplicated without the written consent from ActivMEDIA ROBOTICS, LLC.

The ActivMEDIA ROBOTICS-licensed Saphira libraries that accompany the robot and accessories and which are available for network download by AmigoBot customers are solely owned and copyrighted by SRI International, Inc. The AmigoBot software on disk and on the AmigoBot server FLASH ROM that accompany the robot and accessories and which are available for network download by AmigoBot customers are solely owned and copyrighted by ActivMEDIA ROBOTICS, LLC. AmigoBot 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 the AmigoBot software and hardware without the expressed written consent of ActivMEDIA ROBOTICS is explicitly forbidden.

The various names and logos for products used in this manual are registered trademarks or trademarks of their respective companies. Mention of any third-party hardware or software constitutes neither an endorsement nor a recommendation.

AmigoBot Technical Manual, version 1.3, November 2002.

Federal Communications Commission (FCC) Statement

This equipment has been tested and found to comply with the limits for a class B digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial or residential environment. This equipment generates, uses, and can radiate radio frequency energy, and if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. There is no guarantee that harmful interference won't occur, in which case the user will be required to correct the interference at their own expense. Some possible ways to ameliorate the interference include:

- Reorient or relocate the receiving antenna
- Operate the equipment in a different location
- Connect the equipment to a different power outlet
- Consult with your local dealer or contact support online

Warning

It is essential that only the supplied power and radio units be used.

Any changes or modifications to the equipment not expressly approved by the parties responsible for compliance could void your authority to operate the equipment.

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.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION	7
What is AmigoBot?	7
Hardware	7
Software and Modes of Operation.....	8
AmigoBot Technical Package.....	9
User-Supplied Components / System Requirements.....	9
Additional Resources.....	9
<i>ActivMEDIA ROBOTICS' Software</i>	9
<i>AmigoBot Newsgroup</i>	9
<i>Support</i>	10
CHAPTER 2 SPECIFICATIONS & CONTROLS.....	11
Physical Characteristics	11
Controls, Switches, Indicators, and Sounds	11
<i>Recharge/Power/Battery</i>	11
<i>Reset & Motors/Test Buttons and System/User LEDs</i>	12
<i>Sounds and Volume</i>	12
Motors and Position Encoders.....	13
Sonar	13
<i>Sonar Rate and Sequence</i>	13
<i>Sonar Sensitivity</i>	13
Serial and Accessory Ports.....	14
<i>System/Aux1 Serial Port</i>	14
<i>Control Serial Port</i>	14
<i>Accessory Connector</i>	14
Radio Modems.....	14
Safety Watchdogs and Configuration	15
CHAPTER 3 QUICK START.....	16
Preparative Assembly.....	16
Install ARIA.....	16
AmigoBot Cold Start-Up.....	17
Client-Server Connection	17
<i>A Successful Connection</i>	17
Operating the ARIA Demonstration Client	18
Disconnecting	18
Quickstart Troubleshooting with SRIsim.....	18
<i>Proper Connections</i>	19
<i>SRIsim</i>	19

CHAPTER 4 SELF-TESTS	20
Motors Test	20
Sonar Tests.....	20
Self Wander.....	20
CHAPTER 5 AMIGOBOT OPERATING SYSTEM	21
Communication Packet Protocol	21
<i>Packet Data Types</i>	22
<i>Packet Checksum</i>	22
<i>Packet Errors</i>	23
Server Information Packets.....	23
Client Commands.....	24
<i>Client Command Argument Types</i>	27
Programming AmigOS	27
<i>Synchronization—SYNC</i>	27
<i>Autoconfiguration</i>	27
<i>Opening the Servers—OPEN</i>	28
<i>Keeping the Beat—PULSE</i>	28
<i>Closing the Connection—CLOSE</i>	28
Movement Commands	29
AmigoBot in Motion.....	29
<i>PID Controls</i>	29
<i>Position Integration</i>	30
Sonar	30
Input / Output (I/O)	30
<i>DIGIN and DIGOUT</i>	31
<i>ADSEL</i>	31
<i>Sounds</i>	31
Extended Server Information Packets	31
<i>Packet Processing</i>	31
<i>AUX Serial Packets</i>	31
<i>IOPac and IOREQUEST</i>	32
<i>Configuration Packets</i>	32
<i>Encoder Packets</i>	33
<i>Sound Playlist</i>	34
CHAPTER 6 UPDATING & RECONFIGURING AMIGOS	35
Where to Get AmigOS Software	35
Installing the AmigOS Utilities.....	35
System Mode and Serial Port	35
<i>Updating AmigOS and Sounds with Amigosdl</i>	36
Configuring AmigOS Operating Parameters.....	36
<i>Amigoscf Editor Commands</i>	37

CHAPTER 7 MAINTENANCE & REPAIR	40
Drive Lubrication	40
AmigoBot Batteries	40
<i>Charging the Battery</i>	40
<i>Alternative Battery Chargers</i>	40
Getting Inside	41
Factory Repairs.....	41
 APPENDIX A	 42
System and Control Serial Ports	42
Internal Serial Connector.....	43
Auxiliary Power	43
Motors and Power.....	43
Accessory I/O Expansion Port	43
 APPENDIX B.....	 44
 APPENDIX C	 45
 APPENDIX D	 46
 INDEX.....	 48
 WARRANTY & LIABILITIES	 50

Chapter 1 Introduction

Congratulations on your purchase of an AmigoBot Mobile Robot and welcome to the rapidly growing community of researchers, developers, and enthusiasts of AmigoBot.

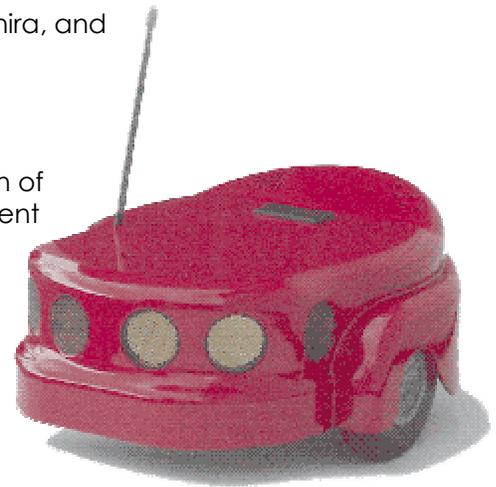
This **AmigoBot Technical Manual** provides both the general and technical details you need to modify and add hardware and software to your AmigoBot Mobile Robot.

We encourage you to also use the companion resources that come with this technical document:

- Personal account for access to the ActivMedia Robotics software and technical documentation library: <http://robots.amigobot.com>
- AmigoBot-users newsgroups
- AmigoBot suite of software, including ARIA, Saphira, and the ActivMedia Robotics Basic Suite

What is AmigoBot?

The AmigoBot Intelligent Mobile Robot is the culmination of many years of mobile-robotics research and development by experts in artificial- and machine-intelligence from around the world. Chief among these contributors is Dr. Kurt Konolige and his team with the Artificial Intelligence Center at SRI International, Inc., a world-class technologies research company once affiliated with Stanford University. AmigoBot is the newest member in the larger family of Pioneer Mobile Robots designed by Dr. Konolige.



AmigoBot is a small, 2-wheel, differential drive, intelligent mobile robot. Like its Pioneer siblings, AmigoBot is truly an off-the-shelf, “plug and play” mobile robot, containing all of the basic components for autonomous sensing and navigation in a real-world environment, including battery power, drive motors and wheels, position / speed encoders, sonar range-finding sensors, and integrated accessories, all managed via an onboard microcontroller and mobile-robot server software.

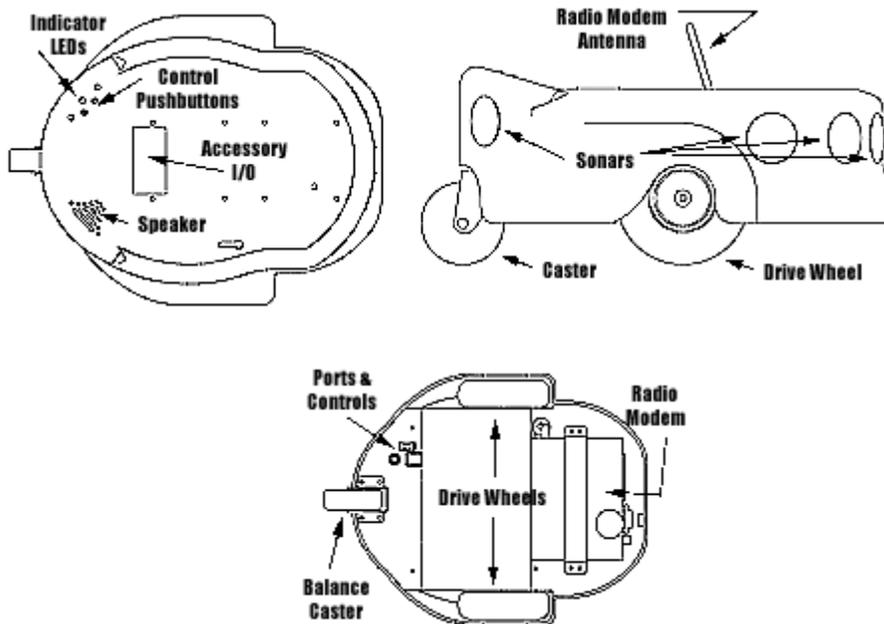
Hardware

The AmigoBot Intelligent Mobile Robot is intended for indoor use in wheelchair-accessible places as are found in most households, school classrooms, nursing homes, hospitals, offices, research labs, and so on. The small (28 x 33 x 13cm), lightweight (3.6Kg with battery), and highly maneuverable (750mm/sec translation; 300°/sec 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 10cm rubber tires are driven by a reversible 12VDC motor. The drive system includes a passive rear caster for balance.

The standard AmigoBot comes with eight range-finding sonars: one on each side, four forward facing and two in the rear for 360-degree sensing coverage. Attached to each drive axle is a high-resolution optical quadrature shaft encoder that provides 9,550 ticks per wheel revolution (30 ticks per millimeter) 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 20-MHz Hitachi H8 microprocessor. The

resident operating system (AmigOS) has 16K RAM and 64K FLASH on-chip memory, with an additional 1M external FLASH.



Your AmigoBot also has a variety of I/O ports for native systems as well as expansion control and power, including two RS232-compatible and one TTL-level serial ports, an 8-bit I/O bus with four chip-select, three address, and read/write lines; six digital I/O ports, four A/D ports; a PWM line, and a variety of 12 and 5VDC power sources.

The system also includes an integrated audio system with speaker and amplifier capable of 9KHz playback of up to 255 (1.7 minutes) pre-recorded sound clips, such as music, voices, and special effects. Alternative sound sets and a downloading tool come with the robot.

Available integrated accessories include AmigoWIREFREE, a 900MHz radio modem pair for wireless control by an offboard client computer and AmigoSURVEILLANCE, a color camera with 2.4GHz A/V radio for live audio/video surveillance applications.

Software and Modes of Operation

The AmigoBot microcontroller comes loaded with AmigOS operating system software that manage all the low-level systems and electronics of the mobile robot. AmigOS is stored in FLASH ROM, as are the robot's systems sounds and operating parameters; all thereby non-volatile, but updateable with special systems-software tools.

AmigOS comes with self-contained programs that operate the robot autonomously without need for other computers or intelligent devices. These Self-Test programs exercise the onboard drives and sensors and have the robot autonomously wander about on its own, navigating around obstacles while performing a simple routine of motions and sounds. (See Chapter 4, *Self Tests*.) Future versions of AmigOS may also let you program your own standalone routines for autonomous performance by AmigoBot.

But we don't recommend that you start learning H8 programming just yet. Rather, AmigoBot prefers to operate by a higher intelligence: smart client applications running on an connected computer whose power and speed is needed to perform complex robotics tasks. Client applications like ActivMedia Robotics' Basic Suite Navigator

communicate with the AmigoBot robot server through the Control serial interface, either directly wired via AmigoLEASH or untethered with the AmigoWIRELESS accessory.

Most people prefer to operate AmigoBot in Client-Server Control Mode, because it gives them quick, easy access to the robot's functionality while working with high-level software on a familiar host computer. Besides the Basic Suite of software including Navigator, as well as the various ARIA demonstration clients, we make client-development environments available so that you, too, may create your own client applications.

And because AmigoOS is virtually identical to the Pioneer, Pioneer 2, PeopleBot, and PowerBot Operating Systems, the clients you program for AmigoBot are easily portable, if not directly applicable, to all ActivMedia intelligent mobile robots. Navigator, for example, operates without modification with the entire line of ActivMedia Mobile Robots, past and future.

Other C/C++ language-based software-development environments for AmigoBot include the very popular Saphira from SRI International, Inc. which is built atop our own ARIA, the foundation application interface for ActivMedia robots.

AmigoBot Technical Package

- CD-ROM containing licensed copies of AmigoBot software and documentation
- Registration and Account Sheet

User-Supplied Components / System Requirements

- Client computer: 486-class or later PC with Microsoft Windows 95-98/ME/2K/NT or Linux operating system.
- AmigoBot Robot with AmigoLEASH serial cable
- Four megabytes of available hard-disk storage

Additional Resources

Every new technical customer gets two additional and valuable resources:

- A private account on our Internet server for downloading ActivMEDIA ROBOTICS' software, updates, and manuals
- Direct access to ActivMEDIA ROBOTICS's technical support team.

ActivMEDIA ROBOTICS' Software

We maintain a 24-hour, seven-day per week Web server where customers can obtain AmigoBot and other ActivMedia Robotics software and support materials:

<http://robots.amigobot.com>

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

AmigoBot Newsgroup

We maintain an email-based newsgroup through which AmigoBot owners share ideas, software, and questions about the robot. To sign up, send an email message to our automated newsgroup server:

```
To: amigobot-users-request@activmedia.com
From: <your return e-mail address goes here>
Subject: <choose one command:>
    help          (returns instructions)
    subscribe
    unsubscribe
```

Our SmartList-based listserver will respond automatically. After you subscribe, send your email comments, suggestions, and questions intended for the worldwide community of AmigoBot users:

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

Access to the amigobot-users newlist 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 AmigoBot.

Support

Have a problem? Can't find the answer in this or any of the accompanying manuals? Do you know a way that we might improve AmigoBot? First consult this manual and check into our online Frequently Asked Questions (FAQ) section on the <http://robots.activmedia.com> server. If you can't find the solution, then contact us:

amigobot-support@activmedia.com

Please include your robot's serial number. Look for it on the underside of your robot. We need it to understand your AmigoBot's configuration to best answer your questions.

Your message goes directly to the AmigoBot technical support team. A staff member will help you or point you to a place where you can find help.

Tell us your AmigoBot's serial number.
We need it.
Really.

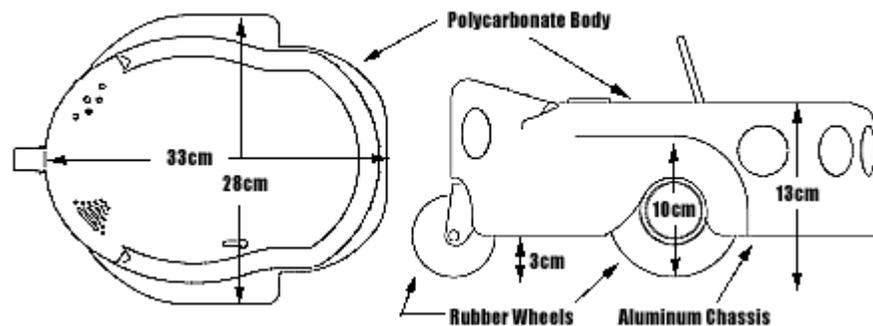
Because this is a support option, not a general-interest newsgroup like `amigobot-users`, we reserve the right to reply only to questions about *problems* with AmigoBot through `amigobot-support`. If you don't hear from us by email within 24 hours, we may have chosen not to respond. Try `amigobot-users`, too.

Chapter 2 Specifications & Controls

AmigoBots may be smaller than most, but they pack an impressive array of intelligent mobile robot capabilities that rival bigger and much more expensive machines. And AmigoBot's modest size lends itself to very safe navigation in tight quarters and cluttered spaces, such as classrooms, laboratories, small offices, and homes.

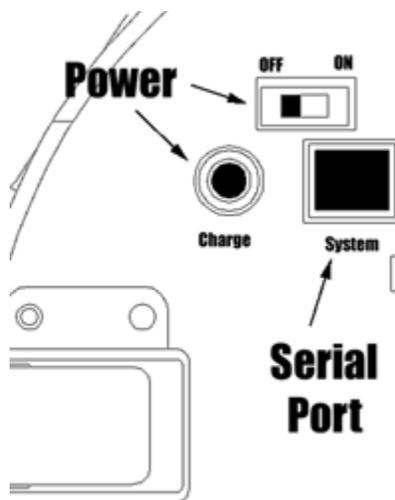
Physical Characteristics

For a complete list AmigoBot's physical and operational specifications, see Appendix D.



Weighing only 3.6Kg (8 lbs. with standard battery), the basic AmigoBot Mobile Robot is lightweight, but its strong polycarbonate body and aluminum chassis make it virtually indestructible. These characteristics also permit AmigoBot to carry an additional payload of up to 1Kg (2.2 lbs.), easily sufficient for available accessories or a sub-compact computer.

Controls, Switches, Indicators, and Sounds



Recharge/Power/Battery

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

Next to the power switch is the Charge port. It provides 12 VDC power to the robot's electronics, motors, and accessories, and recharges the onboard battery. Use the recommended accessory power charger or equivalent.

The standard AmigoBot comes with a single, 12 VDC, 2.2 ampere-hour (26.4 watt-hour) sealed lead/acid battery which supplies ample power for its drives,

electronics, and accessories. Under typical operation with continuous motor activity, the battery provides over 3 hours of service.

You should maintain AmigoBot's batteries in a charged state above 10 VDC. We recommend recharging the battery when it falls below 11VDC (low-batt parameter), even though the robot may continue to operate below 10 VDC. The microcontroller will

Specifications and Controls

sound a warning when the battery voltage falls below that programmed level (see Chapter 6, *Updating and Reconfiguring AmigOS*), and will automatically shut down the motors and any active client-server connection when the battery voltage fall below 10 VDC so to avoid damage.

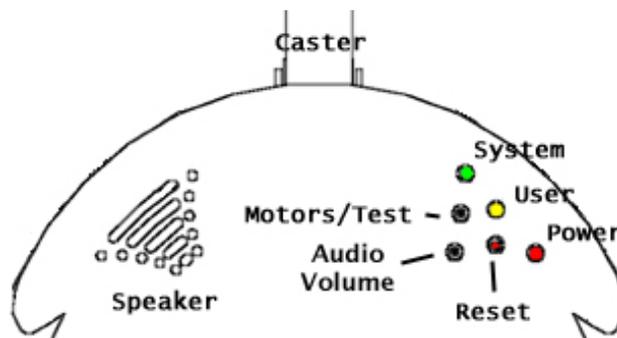
**Disengage the motors when recharging AmigoBot,
but you may continue operating the robot.**

Typical recharge time depends on the charger and the discharge state of the battery. The AmigoBot's standard charger takes overnight (8 hours or more), whereas an 800 ma two-state fast charger option will recharge the battery on 2-4 hours. You may continue to operate AmigoBot while charging its batteries, although that will lengthen the recharge time.

Reset & Motors/Test Buttons and System/User LEDs

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 system. When not connected with a client, press the button once to enable and a second time to begin AmigoBot's self-tests, for instance. While in self-test mode, the button advances the sonar tests. See Chapter 4, *Self-Tests*, for details.



When connected with a client, the black Motors/Test button manually enables and disables the motors, which also can be performed programmatically with the AmigOS command #4, as described in detail in Chapter 5, *AmigOS*.

When pressed in combination, the Motors/Test and Reset button enable system mode on the controller. See Chapter 6, *Updating and Configuring AmigOS*, for details.

The green System and amber User LEDs indicate AmigOS 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 Control serial activity. And when connected, the green System LED flashes rapidly when the motors are disabled, and slowly when the motors are enabled.

Sounds and Volume

The AmigoBot has an onboard audio system capable of reproducing recorded voices, music, and sound effects stored in onboard FLASH ROM. Forty-nine out of the total 255 sounds are dedicated system cues, such as played when the robot is started up or reset, after making a client connection, when stalled, and so on.

The small speaker on the top towards the rear of the robot, across from the control buttons and LEDs is the sound source. Details on programming and playing sounds are in

the Chapter 5, *AmigOS*. A recessed volume control can be turned with a flat-bladed screwdriver to increase or reduce the sound volume.

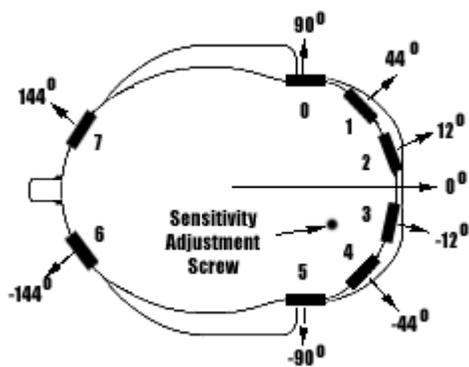
Motors and Position Encoders

AmigoBot's drive system uses high-speed, high-torque, reversible-DC motors. Each front drive motor includes a high-resolution optical quadrature shaft encoder that provides 9,550 ticks per wheel revolution (approx. 30 ticks per millimeter) for precise position and speed sensing and advanced dead-reckoning. The tires are four inches in diameter and made of soft, but firm rubber for good traction and low compressibility.

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.

Sonar Rate and Sequence



The sonar firing rate is 20 Hz (50 milliseconds per sonar) and sensitivity ranges from 10cm (6 inches) to more than 5 meters (16 feet). Objects closer than 10cm are not detected and return an out-of-range value (> 6 meters).

You may control the sonar's firing pattern through software (see Chapter 5, *AmigOS*); the default is clockwise in sequence starting with the side sonar (#0) closest to the left wheel and around to sonar #7 on the back rear panel of the robot.

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.

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.

Serial and Accessory Ports

System/Aux1 Serial Port

Plug your AmigoLEASH cable into the telephone connector-like RJ-11/12 System serial port on the bottom of AmigoBot and its serial adapter connected to your PC to reprogram AmigoOS-related operating software, sounds, and parameters. The port doubles as an auxiliary serial port (AUX1) and is supported in AmigoOS for the attachment of RS232 serial-based accessories.

See Chapters 5 and 6 for details.

Control Serial Port

If your AmigoBot does not have a radio modem, there is a cable that runs from a DSUB-9 serial connector on the bottom of your robot to another telephone connector-like RJ-11/12 serial port inset into the cap that covers the Accessory I/O port. Use the top port with AmigoLEASH to connect with offboard client software in Control Mode, such as Basic Suite Navigator or the ARIA demonstration program.

You cannot connect a control program (“client”) through the System Serial Port.

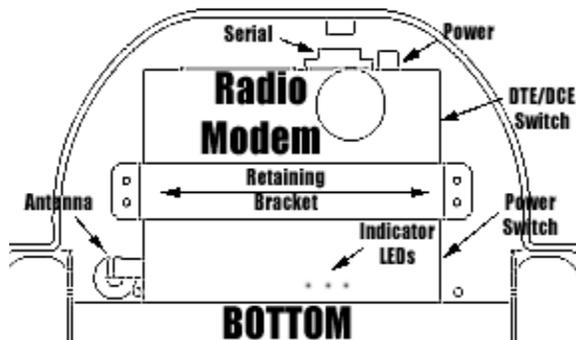
Accessory Connector

Beneath the black rectangular plastic cover near the center of the robot is a 40-position high-density IDC latching connector mounted on the AmigoBot controller board. It supports a variety of accessories through its many I/O ports. See Appendix A, *Ports and Connectors*, for details.

Radio Modems

AmigoBot supports an optional radio modem pair (900 MHz) for wireless operation of the robot: One modem gets attached to the robot and the other to your basestation computer. The robot's modem is mounted on the underside and gets power (5 VDC) and signal (Control serial) via a 9-pin DSUB connector and 2.1mm power plug that come with the robot. The radio's antenna fits up through the body; it's top flexible section unscrews from the main body.

Main power to AmigoBot's radio modem is controlled from the robot's Power switch. Use the pushbutton switch on the side of the modem to individually control its power. A green LED on the modem's face labeled PWR indicates power. When lit, the adjacent red DCD LED indicates a connection with the Host modem of the pair. The DTE LED should remain unlit since the connection is DCE. If lit, slide the switch at the back side of the modem near the power connector to the opposite side so that it is in the DCE position.



The Host radio modem comes with a power module (110 or 220 VAC to 5 VDC) and serial cable with pin adapters. Connect the power module and serial cable to the modem, and the other end of the serial cable into an open serial port on your basestation computer. Operate the power and indicators identically as with the robot's modem.

The radio modems are preconfigured at the factory for use with AmigoBot. For configuration details, see the technical documentation that comes with the modems.

Note that the AmigoBot operates only in client-server Control Mode with the radio modems.

Safety Watchdogs and Configuration

AmigoBot's onboard server software, AmigOS, contains a communications watchdog that will halt motion if communication between a client computer and the robot server is disrupted for a set time interval, normally two seconds (`watchdog`). The robot will automatically resume activity, including motion, as soon as communications are restored.

Also, AmigoBot's server software contains a stall monitor. If the drive exerts a PWM pulse that equals or exceeds a configurable level and the wheels fail to turn (`stallval`), motor power is cut off for a configurable amount of time (`stallwait`). The server software also notifies the client which wheel is stalled. When `stallwait` elapses, motor power resumes and motion continues under server control.

Both these "failsafe" mechanisms help ensure that the robot will not damage objects or be electrically damaged during operation. You may reconfigure the communications, drive current, and stall-wait values to suit your AmigoBot's application. (See Chapter 6, *Updating & Reconfiguring AmigOS*, for details.)

Chapter 3 **Quick Start**

AmigoBot comes fully assembled and ready for action. This chapter describes how to operate the mobile robot with the Saphira demonstration software. (For more details about programming and operating your Pioneer with Saphira, Basic Suite, , see their respective programming manuals.)

The AmigoBot AmigOS servers require a serial communication link to a client. The serial link may be:

- A tether (AmigoLEASH) from the Control serial connector on the top of AmigoBot to a basestation computer
- An optional radio modem pair—one inside AmigoBot and its companion connected to the serial port of a basestation computer.

Preparative Assembly

Out of the box, AmigoBot comes fully assembled, with its batteries installed and fully charged—ready to drive right out of the box. However, you may need to attach an antenna or plug in an accessory that we intentionally leave unattached so as to prevent damage during shipping. Consult any Tech Notes and accessory assembly manuals that may accompany your AmigoBot for final assembly details.

Install ARIA

ARIA client software-development environment, including the ARIA demonstration program and robot simulator, come on CD-ROM with your new robot. ActivMedia Robotics customers also may obtain ARIA and related software and updates from our support website:

`http://robots.activmedia.com`

When installed, ARIA typically requires eight megabytes of hard-disk space.

The Windows version of ARIA is a self-extracting InstallShield® archive. Simply double-click its `.exe` icon and follow the extraction program's instructions. Normally, ARIA is put into a directory named `C:\Program Files\ActivMedia Robotics\ARIA`. The demonstration program and simulator get put into the `bin\` subdirectory. For convenience, you may access all these from the Start Menu's Programs option. The demonstration program's source code and MSVC++ project and workspace files are in the `examples\` subdirectory.

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

`rpm -ihv aria...`

and gets installed in `/usr/local/Aria`. The ARIA demonstration program and simulator get put into the `bin/` subdirectory. The demonstration sources and makefile are in the `examples/` 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 is a terminal application that does not include a GUI, so its programs do not require X-Windows.

AmigoBot Cold Start-Up

Place your AmigoBot on the floor in an open space. Slide the main Power switch on the bottom of the robot to ON. The red power indicator LED on the top of the robot should light and the Power/Reset audio cue should play.

Client-Server Connection

To start the ARIA client demonstration program and connect with AmigoBot, we presume that you have completed the preparatory stages of this chapter by installing ARIA (as needed), by starting and testing the robot, and by connecting the client PC with the AROS controller via a serial link. Now it is time to connect the ARIA demonstration program with your robot.

If you are using radio modems to communicate wirelessly from a desktop PC to the robot, now is a good time to power the units.

Windows users may select the ARIA demo from the start menu, in the ActivMedia Robotics 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:

```
%s ./demo
```

NOTE

By default, the ARIA demo connects with your robot server through your client PC's COM1 (Windows) or /dev/ttyS0 (Linux) serial port.
You must alter the demo source and recompile for a different serial port.

A Successful Connection

ARIA prints out lots of diagnostic text as it negotiates a connection with the robot. If successful, the client requests various AmigOS servers to start their activities, including sonar polling, position integration, and so on. The microcontroller sounds an audible connection cue, and you should hear the robot's sonar ping with a distinctive and repetitive clicking. In addition, the MOTORS/TEST LED should light continuously (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.

Table 1. 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	<ul style="list-style-type: none"> 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	Allows the user to drive and steer the robot via the keyboard or a joystick connected to the computer

Operating the ARIA Demonstration Client

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

The ARIA demo 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, each keypress speeds AmigoBot forward or backward or incrementally changes its direction incrementally. For instance, when turning, it is often useful to press the left- or right-turn key rapidly several times in a row, because the turn increment is small.

The other modes of ARIA demo operation give you access to your robot's various sensors and accessories, including encoders, sonar, I/O port states, and more. Accordingly, use the ARIA demo not only as a demonstration tool, but as a diagnostic one, as well, if you suspect a sensor has failed or is working poorly.

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.

Table 2. Keyboard teleoperation

KEY	ACTION
↑	Increment forward velocity
↓	Decrement forward velocity
←	Incremental left turn
→	Incremental right turn
<i>space</i>	All stop

Disconnecting

When you finish, press the `Esc` key to disconnect the ARIA client from your AmigoBot server and exit the ARIA client 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 `Main Power Switch` to `OFF`.

Quickstart Troubleshooting with SRIsim

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

ATTENTION!

The ARIA client to robot server connection is ***SERIAL*** only.

You may not connect ARIA or other client software with the robot controller over a network TCP/IP link.

Rather, EXPORT the onboard PC (display, keyboard, and mouse) to a remote terminal over the network but run ARIA or other client software locally on the onboard PC.

Proper Connections

Make sure you have ARIA properly installed and that your robot and connections are correct. A common mistake with Linux is not having the proper permissions on the connecting serial port.

Make sure your robot's batteries are fully charged (battery LED green). The robot servers shut down and won't allow a connection at under 11 volts.

The most common problems occur with the serial connection itself, particularly between radio modems. If you are using the onboard PC or radio modems, the serial connection is internal and established at the factory; you should not have problems with those cables. Simply make sure the RADIO's power switch is on, for example.

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 the HOST radio modem of the pair or to the robot's Control serial port. The HOST radio modem should be set to DCE mode and plugged into the serial connector on your PC that serves as COM1 or /dev/ttyS0.

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 a modem OFF, for instance), the client will attempt "Syncing 0" several times and fail. In that case, RESET the robot and check your serial connections. For instance, if you are using radio modems, the DCD lamp on the HOST unit next to your PC should light up. If it doesn't, it means it cannot find the one in the robot.

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

To test for range limits, simply pick up the robot and move it closer to the basestation radio modem. If the robot was out of range, the connection should resume. If not, check to make sure that radio modems were not inadvertently switched OFF.

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 or press the RESET button, you must restart the connection. Turning the Main 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 the.

SRIsim

To verify proper installation of the software, you might run the robot simulator, SRIsim. It is in the same directory as the ARIA demonstration program. Start SRIsim first, then the ARIA demo program. ARIA should successfully connect with the simulator if the software has been installed correctly.

Linux users take note: SRIsim is a GUI program that graphically displays a simulated robot and its world environment in which it operates. Accordingly, you will need to be running X-windows (typically "startx") to use it.

SRIsim looks like a real robot to the ARIA client, so you can operate the demo as you do your own ActivMedia robot. SRIsim includes simulated worlds and different robot profiles which you select from the Files menu, too, so you can see how different robots might navigate in a real or imagined space.

Chapter 4 **Self-Tests**

AmigoOS comes with built-in test routines that exercise AmigoBot's drive motors, sonar, and accessories.

Power up or reset the robot, then press the black Motors/Test button once to hear the AmigoOS version number. You may press the Reset button at any time to disable self-tests.

Place AmigoBot on the floor and have everyone step back
before engaging self-tests.

Another audible cue will alert you that you may have inadvertently engaged the self-tests. Press the black Motors/Test button one again to engage self-tests. If you don't press the button a second time, self-tests automatically cancel after 10 seconds.

Motors Test

The first self-test exercises AmigoBot's drive motors. During this test, the robot is not at all conscious of bystanders. Please have everyone step back and remove any obstacles from within a diameter of about two meters (6-8 feet) around the robot.

The motors self-test begins by engaging the right drive wheel then the left drive wheel, each forward to complete one and one-half full turns counter-clockwise and then clockwise, respectively.

Sonar Tests

Once the motors self-test completes, AmigoBot automatically moves on to test the sonar.

You should hear the distinctive clicking sound as the sonar "ping" individually, each in order clockwise, beginning with the sonar closest to the left wheel (sonar #0). The green System LED blinks at a rate relative to the distance to a target; quicker as a target, such as your hand, approaches.

The sonar are numbered 0 through 7. Press the black Motors/Test button to switch to the next sonar.

Self Wander

After the last sonar test, press the red Reset button to return AmigoBot to its client-server wait state. Or, press the black Motors/Test button to enable Self-Wander mode.

Self-wander is a simple, yet robotically sophisticated program that has AmigoBot drive entirely on its own around the room avoiding obstacles. Every few minutes, the robot will stop, play sounds, spin in place, and then continue on its otherwise aimless journey.

Self-wander is a good demonstration and test of AmigoBot's innate robotic abilities—sensors and onboard intelligence—without having to connect with client—though with very limited capabilities as compared with Saphira or other PC-based robotics client applications.

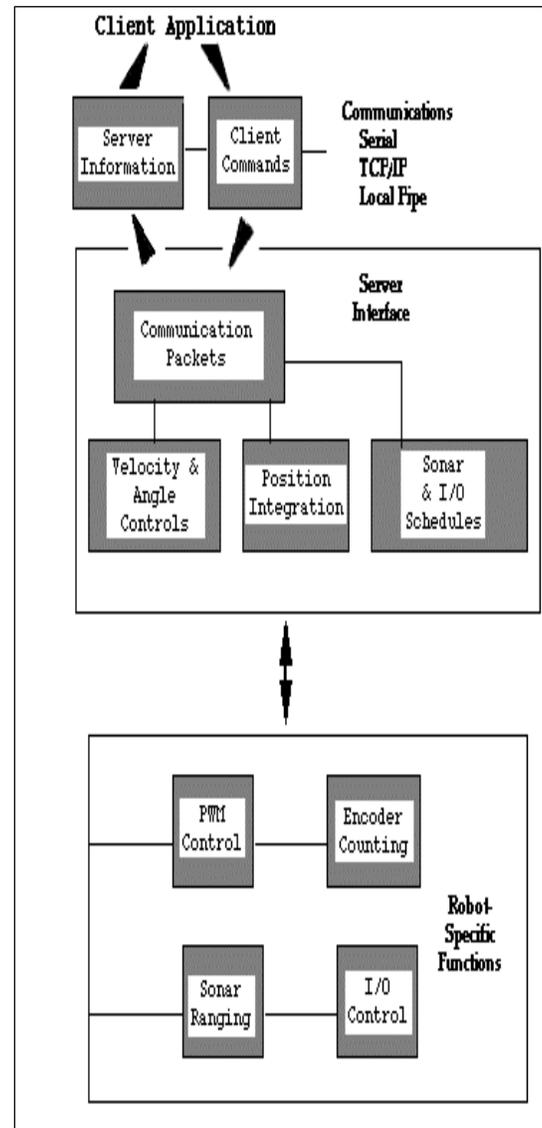
Press the Reset button or switch power OFF at any time to cancel the self-wander test.

Chapter 5 AmigoBot Operating System

All *ActivMEDIA* robots use an intelligent client/server control architecture developed by Dr. Kurt Konolige. In the model, the server works to manage all the low-level details of the mobile robot's systems. These include operating the motors, firing the sonar, collecting sonar and motor encoder data, and so on—all on command from and reporting to a separate client application, such as Saphira and ARIA.

With this client/server architecture, high-level robotics applications developers do not need to know many details about a particular robot server, because the client typically 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 AmigoBot via the AmigoBot Operating System (AmigOS) client/server interface. The same AmigOS functions and commands are supported in the various client-programming libraries that accompany the robot.

Experienced *ActivMedia* robot users already are familiar with AmigOS: It is directly compatible with Pioneer Operating Systems, implementing most of the same commands and protocols. AmigOS, of course, does not support all the PSOS or P2OS-enabled accessories for those respective robots, and extends the servers to add new functionality, such as with sounds.



AmigoBot's client-server architecture

Communication Packet Protocol

AmigOS servers communicate with a client application using special packet protocols: command packets from client to server, and server information packets (SIPs) from the server to client. Both are byte data streams consisting of four main elements: a two-byte header, a one-byte count of the number of command/data bytes, the client command and its arguments or the server information data, and finally, a two-byte checksum.

Main elements of AmigOS communication packet protocol

Component	Bytes	Value	Description
Header	2	0xFA, 0xFB	Packet header; same for client and server
Byte Count	1	N + 2	Number of subsequent data bytes, including checksum word, but not Byte Count. Maximum 200 total bytes.
Data	N	command or SIB	Client command or server information block (SIB; discussed in subsequent sections)
Checksum	2	computed	Packet integrity checksum

Packet Data Types

Client-command and server-information packets use integer (2 byte), word (4 bytes), and string ($n < 200$ bytes) data types. There is no convention for sign; each packet type is interpreted idiosyncratically by the receiver. Negative integers are sign-extended.

AmigOS Communication Packet Data Types

Data Type	Bytes	Order
integer	2	b_0 low byte; b_1 high byte
word	4	b_0 low byte; b_3 high byte
string	up to ~200, length-prefixed	b_0 length of string; b_1 first byte of string

Packet Checksum

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

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

Use the following C-code fragment in your client applications to compute a checksum:

```
int
calc_chksum(unsigned char *ptr)
/* ptr is array of bytes, first is data count */
{
    int n;
    int c = 0;
    n = *(ptr++);
    n -= 2;                /* don't use chksum word */
    while (n > 1) {
        c += (*(ptr)<<8) | *(ptr+1);
        c = c & 0xffff;
        n -= 2;
        ptr += 2;
    }
    if (n > 0) c = c ^ (int)*ptr++;
    return(c);
}
```

Packet Errors

Currently, AmigOS ignores a client command packet whose byte count exceeds 200 or has an erroneous checksum. The client should similarly ignore erroneous server information packets.

AmigOS does not acknowledge receipt of a command packet nor does it have any facility to handle client acknowledgment of a server information packet. Consequently, client/server communications are as reliable as the physical communication link. A cable tether between the robot and client computer provides very reliable links; radio modem-mediated communication is less reliable. Accordingly, when designing client applications that may use radio modems, do not expect to receive every information packet intact, nor can you expect the server to accept every command.

Because of the real-time nature of the client/server interaction, we made a conscious decision to provide an unacknowledged packet interface. Re-transmitting server-information or command packets would serve no useful purpose, because old data would be virtually useless in maintaining responsive client-server interaction.

For some operations, however, the data do not decay as rapidly: Some commands are not overly time-sensitive—for example, those that perform such housekeeping functions as changing the sonar polling sequence. It would be useful to have a reliable packet protocol for these operations, and we are considering this for a future release of the server interface.

In the meantime, 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 the server information packet, client software may detect ignored commands and reissue them until achieving the correct state.

Server Information Packets

Once connected, by AmigOS automatically sends a packet of information over the serial communication line back to the client every 100 milliseconds, depending on the infoCycle setting in the robot FLASH parameters (see Chapter 6, *Updating & Reconfiguring AmigOS*). The standard AmigOS Server Information Packet (SIP) informs the client about a number of the robot's operating parameters and readings, using the order and data types shown in the Table.

AmigOS also supports several additional server information packet types, including an alternative server information packet.

Standard AmigOS Server Information Packet (SIP)

Name	Data Type	Description
Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes + 2 < 201 (0xC9) max.
Status	byte = 0x3S; where S =	Motors status
	sfSTATUSSTOPPED (2)	Motors stopped
	sfSTATUSMOVING (3)	Motors moving
Xpos	unsigned integer (15 ls-bits)	Wheel-encoder integrated coordinates; platform-dependent units; multiply by 0.5083 to convert to millimeters
Ypos	unsigned integer (15 ls-bits)	
Th pos	signed integer	Orientation in platform-dependent units—multiply by 0.001534 for degrees.
L vel	signed integer	Wheel velocities (respective Left and Right) in platform-dependent units; multiply by 0.6154 to convert to mm/sec
R vel	signed integer	

Battery	byte	Battery charge times 10 volts
Bumpers	integer	Motor stall indicators. Bit 0 of the lsbyte is the left wheel stall indicator = 1 if stalled; bit 0 of the msbyte is the right wheel stall.
Control	signed integer	Setpoint of the server's angular position servo—multiply by 0.001534 for degrees
PTU	unsigned integer	bit 0 reflects motors engaged state (1 if engaged) and bit 1 reflects the sonar toggle (1 if on)
Compass	byte	Always 0
Sonar readings	byte	Number of new sonar readings included in information packet; readings follow:
Sonar number	byte	Sonar number
Sonar range	unsigned integer	Sonar reading in millimeters (old AmigOS v1.0 multiply by 0.555)
...rest of the sonar readings...		
Timer	unsigned int	Currently selected analog port number 1-5
Analog	byte	User analog input (0-255=0-5 VDC) reading on selected port
Digin	byte	User digital input; 6 available on b ₀₋₅ ; actual results depend on configuration settings
Digout	byte	User digital output; 6 available on b ₀₋₅ ; actual results depend on configuration settings
Checksum	integer	Checksum (see previous section)

Client Commands

AmigOS implements a structured command format for receiving and responding to directions from a client for control and operation of the robot or its simulator. The number of client commands per second you may send depends on the serial baud rate and average number of data bytes per command. The AmigOS server may not be up to the task of managing a deluge of commands; it reads and processes client commands only once per every 10 ms.

The client must send a command at least once every two seconds or so (`watchdog` parameter; see Chapter 6, *Updating & Reconfiguring AmigOS*); otherwise, the communication watchdog server will stop the robot's onboard drives.

AmigOS client command packet

Component	Bytes	Value	Description
Header	2	0xFA, 0xFB	Packet header; same for client and server
Byte Count	1	N + 2	Number of subsequent command bytes plus checksum, not including Byte Count. Maximum of 200 bytes.
Command Number	1	0 - 255	Client command number; see Table 4-4
Argument Type (command dependent)	1	0x3B or 0x1B or 0x2B	Required data type of command argument: positive integer (<code>sfARGINT</code>), negative integer or absolute value (<code>sfARGNINT</code>), or string (<code>sfARGSTR</code>)
Argument (command dependent)	n	data	Command argument; integer or string
Checksum	2	computed	Packet integrity checksum

AmigOS command set

Command	#	Args	Description	AmigOS	P2OS	AROS
			<i>Before Client Connection</i>			
SYNC0	0	none	Start connection; echoes synchronization commands back to client.	1.0	1.0	1.0
SYNC1	1	none				
SYNC2	2	none				
			<i>After Established Connection</i>			
PULSE	0	none	Client pulse resets watchdog	1.0	1.0	1.0
OPEN	1	none	Starts the controller	1.0	1.0	1.0
CLOSE	2	none	Close client-server connection		1.0	
POLLING	3	string	Set sonar polling sequence	1.0	1.0	1.0
ENABLE	4	int	Enables/disables the motors	1.0	1.0	1.0
SETA	5	signed int	Sets translation acc/deceleration; in mm/sec ²	1.0	1.0	1.0
SETV	6	int	Set maximum translation velocity (mm/sec)	1.0	1.0	1.0
SETO	7	none	Resets server to 0,0,0 origin	1.0	1.0	1.0
SETRV	10	int	Sets maximum rotational velocity; in degrees/sec	1.0	1.0	1.0
VEL	11	int	Move forward (+) or reverse (-) at mm/sec	1.0	1.0	1.0
HEAD	12	int	Turn to absolute heading; 0-359 degrees	1.0	1.0	1.0
DHEAD	13	int	Turn relative to current heading; ± degrees	1.0	1.0	1.0
SAY	15	int,string	Sound duration (20 ms increments)/tone (half-cycle) pairs; <i>int</i> is string length	see SOUND	1.0	1.0
CONFIG	18	int	Request configuration SIP	1.0	1.4	1.0
ENCODER	19	int	Request continuous (>0) or stop sending (=0) encoder SIPs	1.0	1.4	1.0
RVEL	21	signed int	Rotate at ± degrees/sec	1.0	1.0	1.0
DCHEAD	22	int	Colbert relative heading setpoint; ± degrees	1.0	1.0	1.0
SETRA	23	int	Sets rotational (±)de/acceleration in mm/sec ²	1.0	1.0	1.0
SONAR	28	int	Enable (1) or disable (0) the sonars	1.0	1.0	1.0
STOP	29	none	Stops robot (motors remain enabled)	1.0	1.0	1.0
DIGOUT	30	int	Msbits is a byte mask that selects output port(s) for changes; lsbits set (1) or reset (0) the selected port.	–	1.2	1.0
VEL2	32	int	Independent wheel velocities; lsb=right wheel; msb=left wheel; AmigOS in 2cm/sec increments	1.0	1.0	1.0
GRIPPER	33	int	Pioneer Gripper server command. See the Pioneer Gripper manuals for details.	–	1.3	1.0
ADSEL	35	int	Select the A/D port number for analog value in SIP. Selected port reported in SIP Timer value.	1.0	1.2	1.0
GRIPPERVAL	36	int	P2 Gripper.	1.0	1.3	–

IOREQUEST	40	int	Request an IOpac. Set argument=1 for a single packet; >1 for a packet each infoCycle; 0 stop continuous packets	1.2	1.E	1.2
PTUPOS	41	int	Msb is the port number (1-4) and lsb is the pulse width in 100µsec units PSOS or 10µsec units P2OS	–	1.2	–
TTY2	42	string	Send string argument to serial device connected to AUX port on microcontroller	1.0	1.0	1.0
GETAUX	43	int	Request to retrieve 1-200 bytes from the aux serial channel; 0 flushes the aux serial input buffer.	1.0	1.4	1.0
BUMPSTALL	44	int	Stop and register a stall if front (1), rear (2) or either (3) bumping contacted. Off (default) is 0.	–	1.5	–
TCM2	45	int	TCM2 Module commands; see P2 TCM2 Manual for details.	–	1.6	–
E_STOP	55	none	Emergency stop, overrides deceleration	1.0	1.8	1.0
STEP	64	none	Single-step mode (simulator only)	1.0	1.0	1.0
TTY3	66	String	Send string argument out to serial device connected to AUX2 serial port	1.3	1.4	1.0
GETAUX2	67	int	Request to retrieve 1-200 bytes from the AUX2 serial port; 0 flushes the buffer.	1.3	–	1.1
ROTKP	82	int	Change working rotation Proportional PID value (not FLASH default)	1.3	1.M	1.1
ROTKV	83	int	Change working rotation Derivative PID value (not FLASH default)	1.3	1.M	1.1
ROTKI	84	int	Change working rotation Integral PID value (not FLASH default)	1.3	1	1.1
TRANSKP	85	int	Change working translation Proportional PID value (not FLASH default)	1.3	1.M	1.1
TRANSKV	86	int	Change working translation Derivative PID value (not FLASH default)	1.3	1.M	1.1
TRANSKI	87	int	Change working translation Integral PID value (not FLASH default)	1.3	1.M	1.1
REVCOUNT	88	int	Change working differential encoder count (not FLASH default)	1.3	1.M	1.1
SOUND	90	int	Play stored sound	1.0	–	1.0
PLAYLIST	91	int	Request playlist packet for sound number or 0 for all user sounds	1.0	–	1.0
SOUNDTOG	92	int	Mute (0) or enable (1) sounds	1.0	–	1.0

The AmigOS command is comprised of a one-byte command number optionally followed by, if required by the command, a one-byte description of the argument type and the two (integers) or more (strings) byte argument value.

The number of client commands you may send per second depends on the Control serial baud rate, average number of data bytes per command, synchronicity of the communication link, and so on. AmigOS' command processor runs on a ten millisecond interrupt cycle, but the server response speed depends on the command. Typically, limit client commands to a maximum of one every 20 milliseconds or be prepared to recover from lost commands.

Client Command Argument Types

There are three different types of AmigOS client-command arguments: positive integers two bytes long, negative integers two bytes long, and NULL-terminated strings consisting of as many as 196 characters.

The byte order is least-significant byte first. Negative integers are transmitted as their absolute value, unlike information packets, which use sign extension for negative integers; see below. The argument is an integer, a string, or nothing, depending on the command.

Programming AmigOS

You may create your own AmigOS interface, or use the convenience functions available through the various applications development software that comes with your AmigoBot.

Synchronization—SYNC

Before exerting any control, a client application must first establish a connection to the AmigoBot server via its RS-232 Control serial port. Over that established communication link, the client then sends commands to and receives operating information from the server.

When first started, the AmigoBot is in a “wait” state; AmigOS listens for communication packets over its designated port. To establish a connection, the client application must send a series of three synchronization packets through the host communication port—`SYNC0`, `SYNC1`, and `SYNC2`, in succession, and retrieve the server responses.

Specifically, and as examples of the client command protocol, the synchronization sequence of bytes is (in hexadecimal notation):

```

SYNC0:  0xFA, 0xFB, 0x03, 0x00, 0x00, 0x00
SYNC1:  0xFA, 0xFB, 0x03, 0x01, 0x00, 0x01
SYNC2:  0xFA, 0xFB, 0x03, 0x02, 0x00, 0x02

```

AmigOS responds to each client command, forming a succession of identical synchronization packets. The client should listen for the returned packets and only issue the next synchronization packet after it has received the echo.

Autoconfiguration

AmigOS automatically sends configuration information back to the client in the last sync packet (`SYNC2`). After the `SYNC2` byte, there are three NULL-terminated strings that comprise the robot's name, class, and subclass. You may uniquely name your AmigoBot with a special configuration tool we provide. The class and subclass are fixed and cannot be changed by the user.

The class string is `Pioneer`; the subclass depends on your robot model—`amigo` for the AmigoBot. Clients may use these identifying parameters to self-configure their own operating parameters.

Opening the Servers—`OPEN`

Once a communication link is established, the client should then send the `OPEN` command #1 (no argument; `0xFA, 0xFB, 0x03, 0x01, 0x00, 0x01`) which causes the AmigoBot to perform a few housekeeping functions, start its sonar and motor controllers (among other things), listen for client commands, and begin transmitting server information packets.

Note that once connected, AmigoBot'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 black `MOTORS/TEST` button) or send an `ENABLE` client command #4 with an integer argument of 1 (`0xFA, 0xFB, 0x06, 0x04, 0x3B, 0x01, 0x00, 0x05, 0x3B`).

Keeping the Beat—`PULSE`

An AmigOS safety watchdog expects to receive at least one command packet from the client every few seconds. Otherwise, it assumes the client/server connection is broken and shuts down the robot's motors.

It's good practice to send a `PULSE` command #0 (`0xFA, 0xFB, 0x03, 0x00, 0x00, 0x00`) to AmigOS just after opening the servers. And if your client application will be distracted for any appreciable time, periodically issue the `PULSE` client command to let the server know you are indeed alive and well. If the robot shuts down due to lack of communications traffic, it will revive upon receipt of a client command and automatically accelerate to the last-specified speed at the current heading.

Closing the Connection—`CLOSE`

To close a connection, disable the motors and sonar, and reset AmigOS to the wait state, simply issue the client `CLOSE` command number 2 (`0xFA, 0xFB, 0x03, 0x02, 0x00, 0x02`).

With AmigOS versions 1.3 and later, many of the controller's operating parameters return to their FLASH-based default values upon disconnection with the client.¹ For example, if the FLASH default for the maximum velocity is 1000 millimeters per second, and your client uses the `SETV` command #6 to reset the maximum velocity to 500 millimeters per second, the maximum velocity automatically will revert back to 1000 for a subsequent session.

AmigOS motion commands

Rotation	
HEAD	Absolute heading
DHEAD, DCHEAD	Differential heading from control point
SETRA	Rotational (de)acceleration to achieve setpoint
SETRV	Sets rotational velocity for Colbert turn and turnto commands
Translation	
VEL	Forward/back velocity
SETA	Translational (de)acceleration to achieve setpoint
SETV	Velocity for Colbert move command

¹ With earlier versions, the changes persisted between sessions, and reverted to the FLASH defaults only after the controller was reset.

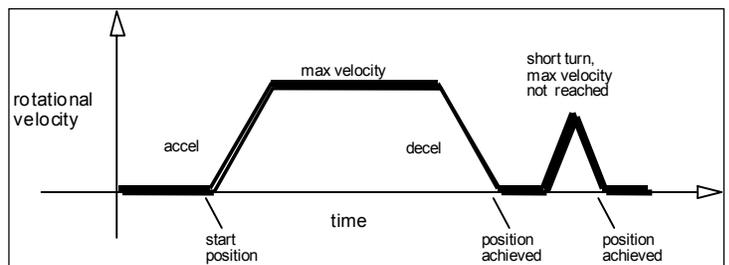
Movement Commands

The AmigOS motors server accepts several different motion commands of two mutually exclusive types: either direct wheel-velocity control or AmigOS motor controls. The robot server automatically abandons any AmigOS translational or rotational setpoints and switches to direct wheel-velocity control mode when it receives a SETVEL2 command. Any other motion command makes AmigOS abandon direct wheel-velocity control. For example, if the AmigOS is in two-wheel velocity mode and is given a HEAD command, it disables two-wheel velocity mode and starts controlling the heading and velocity of the robot.

When in AmigOS motion control (recommended), translation and rotation operate independently. AmigOS will try to make the robot achieve the desired velocity and heading as soon as the commands are received, using its internal acceleration and deceleration managers, which default values you may set and change on the fly (SETRA and SETA).

AmigoBot in Motion

When AmigOS receives a translation or rotation command, it acc(de)elerates the AmigoBot at the SETA or SETRA rate you program, or the rates preset in the AmigOS configuration parameters. Rotational headings are achieved by a trapezoidal velocity function which is re-computed each time a new heading command is received, making on-the-fly orientation changes possible.



Trapezoidal turning velocity profile

Note that you may override deceleration with the emergency stop (E_STOP) command number 55. Accordingly, with E_STOP, the robot brakes to zero translational and rotational velocities with very high deceleration and remains stopped until it receives a subsequent translational or rotational velocity command from the client.

PID Controls

The AmigOS drive servers use a common Proportional-Integral-Derivative (PID) control system to adjust the PWM pulse width at the motor drivers and subsequent power to the motors. The motor-duty cycle is 200 microseconds; pulse-width is proportional 0-255 for 0-100% of the duty cycle.

The AROS drive servers recalculate and adjust your robot's trajectory and speed every ten milliseconds based on feedback from the wheel encoders.

The default PID values for translation and rotation and maximum PWM are stored as FLASH parameters in your robot's microcontroller and may be changed. You also may temporarily update the PID values with the AmigOS client commands 84 through 87. On-the-fly changes persist until the controller is reset. The translational PID values apply to independent wheel-velocity mode.

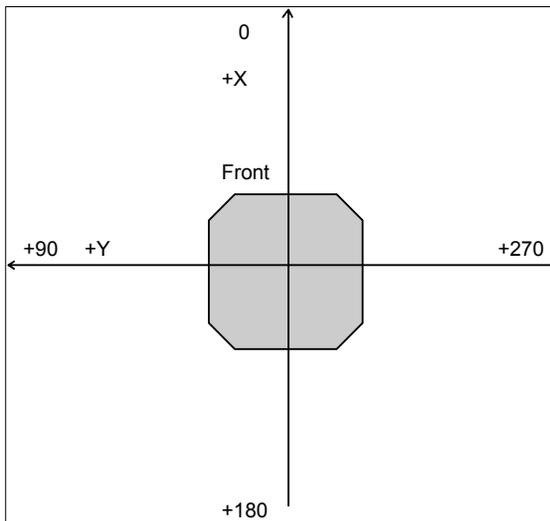
The P term value K_p increases the overall gain of the system by amplifying the position error. Large gains will have a tendency to overshoot the velocity goal; small gains will limit the overshoot but cause the system to become sluggish.

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.

The I Term Ki 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 Ki 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.

Position Integration



Internal coordinate system

AmigoBot keeps track of its position and orientation based on dead-reckoning from wheel motion, which is an *internal coordinate position*. 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 several meters and one revolution, 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.

The orientation commands HEAD (#12) and DHEAD (#13) turn the robot with respect to its internal dead-reckoned angle. On start-up, the robot is at the origin (0,0), pointing toward the positive x-axis at 0 degrees. Absolute angles vary between 0 and 360 degrees. As the robot moves, it will update this internal position based on dead-reckoning.

You may reset the internal coordinates to 0,0,0 with the SETO command #7.

Sonar

When connected and opened, the AmigOS sonar server begins firing AmigoBot's sonar in the predefined default sequence clockwise beginning with the sonar closest to the left wheel (sonar #0). AmigOS also begins sending the sonar ranging results (millimeters) to the client via the server-information packet. Use the SONAR command #28 to enable (argument is "1") or disable (argument is "0") the sonar pinging.

Input / Output (I/O)

Your AmigoBot comes with a number of I/O ports that you may use for some of its accessories and for your own custom attachments. See Appendix A, *Ports and Connections* for port locations on the AmigoBot microcontroller. The various port states and reading appear in the SIP and may be manipulated with AmigOS client commands.

DIGIN and DIGOUT

With the AmigoBot configuration tool (see next Chapter), you decide which of six I/O ports (IO_0-5) are inputs and which are outputs. The port states are mapped by AmigOS in the standard SIP as corresponding bits in the standard SIP's `Digin` and `Digout` bytes.

Digital output ports get set through the AmigOS DIGOUT command #30 with a 2-byte integer argument. The first byte contains a mask for the bits you want to change; the second byte in the argument contains the desired state for the selected bits.

ADSEL

Use the client command ADSEL to select the A/D port that is to appear in the AmigOS SIP analog value. The default port is #1 out of four total. The AmigOS SIP reports the currently selected analog input port number as the `Timer` value.

Sounds

The AmigoBot comes with an onboard audio amplifier and FLASH-stored sounds. To play an sound, send the SOUND command with the sound number (1-255) argument. Use the SOUNDTOG command to disable (argument of 0) or enable (argument = 1) all sounds.

Extended Server Information Packets

AmigOS has several extended Server Information Packets to better support the AmigoBot robotics community. On request from the client by a related AmigOS command, the AmigOS server will package and send one or a continuous stream of packet types to the client over the Control serial communication line.

Packet Processing

It is up to the client to handle all packet types. Extended packets get intermingled with the standard SIP that AmigOS sends every 100 or 50 milliseconds. Please consult the respective client application programming manuals for details.

The AmigOS extended packets are packaged similarly to the standard Server Information Packet, including header (0xFA, 0xFB) and the checksum. The difference, besides the number and included data, of course, is a different packet type (`Status` byte). Accordingly, the client processor should simply examine the packet type byte and process each packet accordingly.

A sample packet processor is included with the Saphira distribution—`packet.c` in the `apps/` directory.

AUX Serial Packets

Use the AmigOS command number TTY2 42 to send a string (command argument) from the client through the Control serial port and out the AUX1 serial port, which beginning with AmigOS version 1.3, is shared with the System serial port on the RJ-11 connector on the underside of your AmigoBot.² Use the client command number 43 (GETAUX) to request feedback from the AUX1 serial port. It tells AmigOS to retrieve a given number of bytes (command argument value) from the serial device attached to that RS232-compatible AUX port, and to package and send those incoming data in a special SERAUXpac (type 176; 0xB0) SIP back to the client via the common HOST serial port.

² Previously, only one AUX port was implemented and it was the TTL-level one present on the expansion I/O connector.

Similarly, beginning with version 1.3, use the AmigOS TTY3 command #66, GETAUX2 command #67, and corresponding SERAUX2 (type = 0xB8) extended SIP to send a string to, and request and receive data bytes back through the TTL-level second auxiliary serial port on your AmigoBot's expansion IO connector.

SERAUX and SERAUX2 Server Information Packets contents

Name	Data Type	Description
Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes = n data +3.
Type	byte	0xB0=SERAUX; 0xB8=SERAUX2
Data	string	N data bytes as requested in command
Checksum	integer	Checksum for packet integrity

AmigOS maintains a circular buffer for incoming serial data from the AUX1 and AUX2 ports and returns successive portions depending on the number of bytes you request via the GETAUX(2) command, up to 200 total at a time. AmigOS waits to collect that number of incoming AUX-port serial bytes before sending the packet to the client. Use the GETAUX(2) command with a zero argument to flush the AmigOS circular buffer and reset its pointers.

IOpac and IOREQUEST

AmigOS 1.2 introduced a new server information packet in which your AmigoBot robot controller reports all of its I/O-connected sensor input and output values in a single cycle.

IOpac packet contents

Header	integer	Exactly 0xFA, 0xFB
Count	byte	Number of data bytes + 2
Type	byte	Packet type = 0xF0
N _i	byte	Number of digital input bytes
Digin	N _i bytes	Digital input bytes
N _o	byte	Number of digital output bytes
Digout	N _o bytes	Digital output bytes
N _a	byte	Number of A/D values
A/D	N _a ints	A/D values 1-N _a ; 0-2047 (12-bit resolution) = 0-5 VDC
Checksum	integer	Computed checksum

Your client software must explicitly request IO packets; they normally are disabled. Use the AmigOS IOREQUEST command #40 with an argument value of 0, 1, or 2. The argument 1 requests a single packet to be sent within the next cycle. The request argument value 2 tells AmigOS to send IO packets continuously, at approximately one per cycle, depending on serial port speed and other pending SIPs. Use the IOREQUEST argument value 0 to stop continuous packets.

The common AmigOS IOpac contains one digital input (digin) , one digital output byte (digout), and five analog-to-digital values corresponding to the ports AN1-5. Unlike the standard SIP, which contains a byte value for the selected analog port, analog values in the IOpac are integers, with resolution to 12 bits. If no electronics are attached to the digital input or analog I/O ports, their values can vary.

Configuration Packets

Send the CONFIG command number 18 with any integer argument to request AmigOS send back a special server information packet containing the robot's operational parameters. The CONFIG SIP packet type is 32 (0x20).

CONFIGpac contents

label	DATA	DESCRIPTION
header	int	Common packet header = 0xfAFB
Type	byte	IDs ENCODERpac = 0x20
Byte count	byte	Number of following data bytes
Robot Type	str	Typically "Pioneer"
Subtype	str	Identifies the <i>ActivMedia</i> robot model; "amigo", for example.
Serialnum	str	Serial number for the robot.
4Mots	Byte	Antiquated (=1 if AT with P2OS)
RotVelTop	int	Maximum rotational velocity; deg/sec
TransVelTop	int	Maximum translation speed; mm/sec
RotAccTop	int	Maximum rotation (de)acceleration; deg/sec ²
TransAccTop	int	Maximum translational (de)acceleration; mm/sec ²
PwmMax	int	Maximum motor PWM (255=fully on).
Name	str	Unique name given to your robot.
InfoCycle	byte	Server information packet cycle time (0-100ms; 1=50ms)
HostBaud	byte	Baud rate for client-server Control serial port: 0=9.6k, 1=19.2k, 2=38.4k
Aux1Baud	byte	Baud rate for AUX1 serial port 1; see HostBaud
gripper	int	Always 0 since AmigoBot has no gripper accessory
front sonar	int	Always 1 since AmigoBot has 1 sonar array
aux sonar	byte	0; only one sonar array
LowBattery	int	In 1/10 volts; alarm activated when battery charge falls below this value.
RevCount	int	Current number of differential encoder ticks for a 360 degree revolution of the robot.
WatchDog	int	Ms time before robot automatically stops if it has not received a command from a client. Restarts on restoration of connection.
P2Mpacs	byte	1 enables alternative SIP.
StallVal	int	Maximum PWM before stall. If > PwmMax, never.
StallCount	int	Ms time after a stall for recovery. Motors not engaged during this time.
JoyVel	int	0; AMigobot doesn't have a direct joystick port
JoyRVel	int	ditto
RotVelMax	int	Current max rotational speed; deg/sec.
TransVelMax	int	Current max translational speed; mm/sec.
RotAcc	int	Current rotational acceleration; deg/sec ²
RotDecel	int	Current rotational 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 translational acceleration; mm/sec ²
TransDecel	int	Current translational 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

Encoder Packets

By issuing the ENCODER command number 19 with a non-zero integer argument, you initiate a continuous stream of ENCODERpac (type 144; 0x90) SIPs. One ENCODERpac is sent every 100 or 50 milliseconds, depending on the standard packet cycle rate (sInfoCycle) Discontinue the packets by sending the ENCODER command number 19 with the argument = 0.

ENCODER Server Information Packet

Name	Data Type	Description
Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes = 11.
Type	byte	0x90
Left Encoder	integer	Least significant portion and the most significant portion comprise the current 4-byte raw encoder count from the left drive wheel
Right Encoder	integer	Least significant portion and the most significant portion comprise the current 4-byte raw encoder count from the right drive wheel
Checksum	integer	Checksum for packet integrity

Sound Playlist

Sound files stored in FLASH on the AmigoBot microcontroller contain a descriptive header called the Playlist which includes a sound or sound group name³ with data address and length. AmigOS and clients use the first 49 sounds (1-49) for system related events and activities. Sounds 49-255 are general purpose.

Use The PLAYLIST command with the sound number as argument to retrieve the PLAYLISTpac type (0xD0; 208) for a single sound or argument 0 to retrieve the playlist for all user sounds. In the latter case, the first playlist request reports the number of sounds and then sends a playlist information packet for each sound every 100 or 50 (SinfoCycle) milliseconds.

PLAYLIST Information Packet

Name	Data Type	Description
Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes = n data +3.
Type	byte	0xD0
Sound number	byte	Sound number in playlist or number of sounds if first response packet when getting all user playlist.
Sound toggle	byte	Sound on (1) or off (0).
Name	16 byte string	Sound name in ASCII may be empty or may not be null-terminated. If tilde (~) prefix, is a sound group name.
Offset	4 bytes	Address offset to sound data; 0 if a group name or empty sound.
Length	4 bytes	Length of sound data; 0 if a group name or empty sound.
Checksum	integer	Checksum for packet integrity

³ Name only with tilde (~) prefix designates the following sounds as members of a group of sounds. There is no associated sound data with the group playlist entry.

Chapter 6 Updating & Reconfiguring AmigOS

The AmigOS server software and its set of operating parameters get stored on the AmigoBot microcontroller's flash ROM. With special download and configuration utilities, you may change and update the flash ROM image without physically replacing any hardware.

Where to Get AmigOS Software

Your AmigOS comes preinstalled with the latest version of AmigOS. Thereafter, stay tuned to the `amigobot-users` newsgroup, or periodically visit our support website to obtain the latest AmigOS and related documentation:

<http://robots.amigobot.com>

Installing the AmigOS Utilities

The AmigOS utilities come with this technical document or can be downloaded from the support website as a separate package. Install the version that matches your client computer's environment. For example, use `AmigOS1_3.tgz` for RedHat Linux or `AmigOS1_3.EXE` with Microsoft Windows 32-bit systems. We distribute these as compressed archives containing all the programs and accessory files you need to perform the AmigOS upgrade and to set your AmigoBot's configuration parameters.

The `.EXE` distribution is a self-extracting, self-installing WinZIP package: Simply follow the on-screen directions. For the Linux/UNIX versions, uncompress and untar the distribution using the appropriate system utilities.

For example with the Linux version, the command is:

```
% tar -zxvf AmigOS1_0.tgz
```

In all case, the archive extracts into an `AmigOS` directory in the selected path and stores the AmigOS utilities and images there.

System Mode and Serial Port

Changes and updates to your AmigoBot's programs and parameters are done through the special System serial port on the bottom of the robot and through the controller's System servers and communication port.

Connect an AmigoLEASH or equivalent serial cable between the System serial port on the bottom of AmigoBot to a serial port on your host computers.

Start up or Reset your AmigoBot. After it has finished initializing, place it into System mode: While pressing and holding the black Motors/Test pushbutton, press and release the red Reset button. Then release the black Motors/Test button.

The robot should not reset or start Self Tests. If it does, reset and try again. When in System Mode, AmigoBot's amber User LED stays lit and the green System LED flashes twice as fast as when in client-connection waiting mode.

While in System Mode you may start and quit any of the system-related utilities without resetting the controller or re-establishing System Mode as you may have done with a Pioneer Mobile Robot.

Updating AmigOS and Sounds with Amigosdl

AmigOS software, including sound files, get distributed as Intel Hex files, which encode both data and addressing for the system. Use the `amigosdl(.exe)` utility to download a fresh AmigOS or change your sound file data.

With Linux/UNIX systems, enter the `AmigOS/` directory and execute `amigosdl`, using the following arguments:

```
% ./amigosdl [pathname].hex <comm-port>
```

The `[pathname]` for the `Hex` image file is required and may be a full or relative pathname. The file `AmigOS1_0.hex`, for example, is the current AmigOS version 1.0 image in the `AmigOS` directory.

The `comm-port` argument is optional. It lets you specify the serial communication port that connects `amigosdl` with the AmigoBot System serial port. For Linux/UNIX systems, the default is `/dev/ttyS0`. To communicate through `/dev/ttyS3`, for example, use:

```
% ./amigosdl newsounds.hex /dev/ttyS3
```

For Microsoft Windows systems, you must specify the required AmigOS file, so you cannot simply double-click on the program icon from the Windows desktop. Instead, execute `amigosdl.exe` and pass it the proper arguments from the MS-DOS Prompt program, which normally resides in the `Programs` section of the `Start` menu. The default serial port is `COM1`.

Configuring AmigOS Operating Parameters

The program `amigoscf(.exe)` is the way you view and change your AmigoBot's identity and operating parameters.

As with `amigosdl`, connect with AmigoBot's System serial port on the bottom of the robot and enable System Mode on the robot, if you haven't done so already.

Find and execute `amigoscf(.exe)` in the `AmigOS` directory of your AmigOS utilities distribution.

```
% ./amigoscf <-p comm-port || -n>
```

The program accepts a couple different command options than `amigosdl`: Use the `-p` flag to specify a serial communication port name other than the `COM1` or `/dev/ttyS0` one (`-p com2, for example`). Or use the `-n` flag to start `amigoscf` without having to make a connection with AmigoBot. The latter mode is useful if you want to change your disk file-based parameters for various configurations of AmigoBot. Also, if you use the default `COM1` on your Win32 PC, you may launch `amigoscf.exe` directly from the desktop, rather than executing the program from the MS-DOS Prompt window.

On startup (after power on or Reset), AmigOS reads a set of operating parameters from its FLASH ROM and uses these values if and until you override them with explicit AmigOS commands. For instance, a default maximum velocity is stored in FLASH (`TransVelMax`) which value is used by AmigOS when receiving a Colbert `move` command or other "goto position" client command. The robot accelerates to `TransVelMax` and runs at that velocity until it nears and then decelerates as it reaches its destination.

When started, `amigoscf` retrieves from the microcontroller the current identifying and operating parameters that AmigOS uses for your AmigoBot. Some of the parameters, "Constants" in the Table, cannot be changed. The others, "Variables", are the identifying and operating parameters that you may edit and save in

FLASH with `amigoscf`. Your changes are not written to the controller's FLASH until you choose to explicitly "save" them. You also may save a copy of the variable parameters in a disk-based file for later recovery.

Amigoscf Editor Commands

To view the list of current AmigOS constants or variables, type '`c`' or '`v`', respectively, followed by a return (Enter). Similarly, type '?' or 'help' to see a list of `amigoscf` commands.

To see a parameter's current value individually, type its keyword alone. To change an AmigOS variable parameter's value, type its keyword followed by the replacement value. That value may be a string (no quotes) or a decimal or hexadecimal ("0xN") number. For example, to change the `watchdog` timeout to last for four seconds, type:

```
> watchdog 4000
or:
> watchdog 0xfa0
```

The critical operating parameters for AmigoBot are `revcount` and the PID control parameters. If you get them wrong, your robot won't run properly. Note, too, that your `amigoscf`-edited parameters are *not* used by AmigOS unless and until you 'save' them to FLASH. And, too, you may over-ride many of these parameters with respective AmigOS commands from the client.

Amigoscf Commands

Command	Description
<code>keyword <value></code>	Alone, keyword displays current, edited value. Add a value argument to change current value.
<code>c</code> or <code>constants</code>	Display AmigOS constant values. You cannot change these.
<code>v</code> or <code>variables</code>	Display current, edited AmigOS operational values; may be different than values currently stored in FLASH.
<code>r</code> or <code>restore</code> <code><pathname></code>	Restores edited variables to values currently stored in FLASH or from a file, if filename argument included.
<code>save <pathname></code>	Saves current edited values to FLASH or saves current edited values to disk for later reference.
<code>q</code> or <code>quit</code>	Exits <code>amigoscf</code> .
<code>?</code> or <code>help</code>	Displays commands and descriptions.

Saving and Restoring Parameters

The AmigOS configuration program lets you save and restore whole configuration sets from stored files. This lets you easily configure AmigoBot for the various different environments, as well as maintain a record of your original and test parameters.

To save your current configuration to a disk file, get connected with the AmigoBot with `amigoscf` as described earlier. This loads the current operating parameters into the configuration editor. Then simply provide an argument to the save command consisting of the pathname for the configuration file.

AmigoBot Configuration parameters (AmigoS 1.0)

KEYWORD	Type	Default	Description
CONSTANTS			<i>Cannot be changed using p2oscf</i>
Type	string	Pioneer	String identifies the robot as a Pioneer type and is included in the SYNC2 connection return packet along with Subtype and Name.
Subtype	string	Amigo	Identifies the AmigoBot model.
Serial	string	factory set	Serial number for the AmigoBot.
RotVelTop	integer	360	Maximum allowable rotational velocity in degrees per second
TransVelTop	integer	2200	Maximum allowable speed in millimeters per second
RotAccTop	integer	360	Maximum allowable rotational (de)acceleration in degrees per second
TransAccTop	integer	4000	Maximum allowable translational (de)acceleration; millimeters per second
PwmMax	integer	255	Maximum motor pulse period (255=fully on).
VARIABLES			<i>Parameters that you may change with p2oscf</i>
Name	string	not_set	Unique name you may give your AmigoBot. Besides its ownership value, this parameter gets passed to a connecting client as the first argument in the SYNC2 packet, therefore useful for differentiating among multiple AmigoBots. Maximum of 20 characters; no intervening spaces.
SInfoCycle	byte	0	Standard SIP communication cycle time: 0=100, 1 = 50 milliseconds.
HostBaud	byte	0	Baud rate for Control(client) serial port connection. 0=9600, 1=19200, 2=38400 bps.
Aux1Baud	byte	0	Baud rate for (AUX1) serial port; values as for HostBaud
Aux2Baud	byte	0	Baud rate for (AUX2) serial port; values as for HostBaud
io0	byte	0	Port IO_0 is an input port if 0; output if 1
io1	byte	0	Port IO_1 is an input port if 0; output if 1
io2	byte	0	Port IO_2 is an input port if 0; output if 1
io3	byte	0	Port IO_3 is an input port if 0; output if 1
io4	byte	0	Port IO_4 is an input port if 0; output if 1
io5	byte	0	Port IO_5 is an input port if 0; output if 1
LowBattery	integer	110	In 1/10 volts; microcontroller alarm activated when battery charge falls below this value. Automatic shutdown always at < 10V.
WatchDog	integer	2000	Milliseconds time before robot automatically stops if it has not received a command from a client. Restarts on restoration of connection with client.
RevCount	integer	23310	The number of encoder ticks for a 360 degree revolution of the robot. Reset this parameter to a number that best reflects the characteristics of your robot in a particular environment. See <i>revcountcal</i> utility.
Mpacs	byte	0	'1' enables new, extended AmigoOS server information packet.
StallVal	integer	255	Maximum PWM before stall; either or both motors. If > PwmMax, never stalls.
StallCount	integer	100	Milliseconds after a stall for recovery. Motors not engaged during this time.
RotVelMax	integer	200	Maximum velocity for completion of a Colbert or similar rotation.
TransVelMax	integer	300	Maximum velocity for completion of a Colbert or similar translation.
RotAcc	integer	50	Rotational acceleration in degrees per second
RotDecel	integer	50	Rotational deceleration in degrees per second
RotKp	integer	30	Proportional PID parameter for responsiveness of the drive system. Lower values make a slower, less-responsive system; higher values make the robot "zippier", but can lead to overshoot and oscillation.
RotKv	integer	60	Differential PID dampens oscillation and overshoot. Increasing values gives better control oscillation and overshoot, but they also make the robot's movements more sluggish.
RotKi	integer	0	Integral PID 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.
TransAcc	integer	300	Translational acceleration in degrees per second
TransDecel	integer	300	Translational deceleration in degrees per second
TransKp	integer	40	<i>see RotKp</i>
TransKv	integer	80	<i>see RotKv</i>
TransKi	integer	0	<i>see RotKi</i>

For example, to save your current configuration:

> **save C:\AmigOS\myAmigoBot**

The command does not change the working configuration in any way and does not include or save the configuration constants.

Use `amigoscf` to restore operating parameters either from the robot or from a saved parameters files on disk. The AmigOS utilities come with the default configuration in a file called `AmigoBot.cf`. Restored parameters overwrite the temporary parameters in the `amigoscf` program and do not take effect on the robot until `save`'d onto its FLASH.

You may edit the file-restored parameters just as you edit those retrieved from the robot. And you may save those edited parameters back out the same file or a different one, using the `amigoscf save` command.

Chapter 7 **Maintenance & Repair**

Your AmigoBot Intelligent Mobile Robot is built to last a lifetime and requires little maintenance.

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.

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 the AmigoBot-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:

AmigoBot-support@activmedia.com
(603) 924-2184 fax
(603) 924-9100 (voice)

In the body of your e-mail or fax message, provide your robot's serial number found on its underside, 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.

We are not responsible for shipping damage or loss.

Appendix A

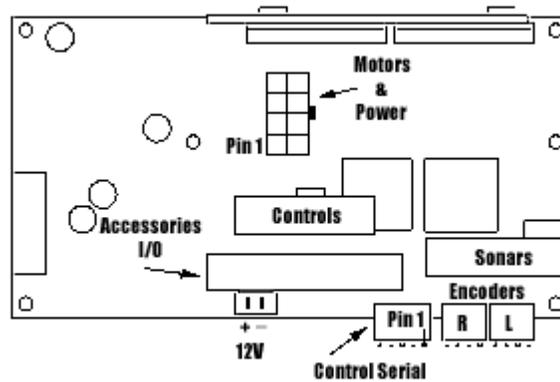
Controller Ports & Connections

This Appendix contains pinout and electrical specifications for the external and internal ports and connectors on the AmigoBot microcontroller board.

13	11	9	7	5	3	1
14	12	10	8	6	4	2

Common IDC connector pinouts

AmigoBot Controller Connectors



System and Control Serial Ports

Use the System RJ-11/12 connector on the bottom of the AmigoBot for connections with AmigoOS utilities, such as `amigoscf` and `amigosd1`, but not for client connections. An identical connector attached to the Control serial port of the microcontroller should be used for client-software control connections.



Only three pins of the Control and System serial ports are active: RS232-compatible signals (RxD) and TxD with reference GND. Careful when using a common 4-wire telephone cable with the System port—the connector pins get reversed. AmigoLEASH has a 9-pin DSUB adapter which RJ-11/12 connector uses a common telephone cable for connection.

The AmigoBot operates at 9,600, 19,200, or 38,400 bits per second; eight data bits; one stop bit; with no parity or hardware handshaking.

Control-Mode Serial/ Radio Modem Connectors

Internal Pin #	Signal Connection	Modem 9-pin Female
1	Power Gnd	5
2	5 VDC	-
3	Signal Gnd	-
4	TxD	3
5	RxD	2
6	RxD	-
7	TxD	-
8	RxD	-

Internal Serial Connector

One 8-position latch-lock connector on the AmigoBot microcontroller supports Control Mode RS-232 serial communications and power for a radio modem. If the radio is not installed, an RJ11/12 connector provides connection to a PC via AmigoLEASH. A null-modem adaptor is needed to connect a common serial cable to the DSUB9 female connector that normally attaches to the modem.

Auxiliary Power

A 2-position latch-lock header on the AmigoBot controller supplies battery (~12VDC unconditioned) power for accessories, such as the AmigoSURVEILLANCE camera, microphone, and A/V transmitter.

Motors and Power

An eight-position mini-fit junior connector provides power from the battery to the microcontroller and to the motors.

AmigoBot Motor/Power Connector

Pin #	Connection	Pin #	Connection
4	R Motor –	5	BATT-
3	R Motor +	6	BATT+
2	L Motor –	7	BATT-
1	L Motor +	8	BATT+

Accessory I/O Expansion Port

A 40-pin high-density IDC latching header on the AmigoBot microcontroller provides digital, analog, and power ports for accessory connections.

I/O Ports on Accessories Connector

Pin #	Label	Description	Pin #	Label	Description
1	12V	Battery power unconditioned	2	GND	Battery power ground
3	12V		4	GND	
5	12V		6	GND	
7	IO_5	Digital I/O	8	IO_5	Digital I/O
9	IO_0	Digital I/O	10	IO_1	Digital I/O
11	Gnd	Signal Ground	12	IO_2	Digital I/O
13	PWM	User PWM	14	IO_3	Digital I/O
15	Gnd	Signal Ground	16	TxE0	Control serial RS232
17	D0	Data Bus	18	RxE0	Control serial RS232
19	D1		20	Tx2	AUX serial TTL
21	D2		22	Rx2	AUX serial TTL
23	D3		24	AN0	A/D
25	D4		26	AN1	
27	D5		28	AN2	
29	D6		30	AN3	
31	D7		32	RD	Bus read
33	CS2	Chip Selects	34	WR	Bus write
35	CS3		36	A0	Bus Address
37	CS4		38	A1	
39	CS5		40	A2	

Appendix B

ARIA/Saphira AmigoBot Parameters File

```
;;
;; Parameters for the AmigoBot
;;

AngleConvFactor 0.001534 ; radians per angular unit (2PI/4096)
DistConvFactor 0.5083 ; mm
VelConvFactor 0.6154 ; mm/sec
RobotRadius 180.0 ; radius in mm
RobotDiagonal 120.0 ; half-height to diagonal of display octogon
Holonomic 1 ; turns in own radius
MaxRVelocity 300.0 ; degrees per second
MaxVelocity 1000.0 ; mm per second
RangeConvFactor 1.000 ; sonar range mm
DiffConvFactor 0.011 ; rotational velocity convert to deg/sec

;;
;; Robot class, subclass
;;
Class Pioneer
Subclass amigo
SonarNum 8 ; total sonars

;;
;; Sonar parameters
;; SonarNum is number of sonars
;; SonarUnit I X Y TH is unit I (0 to N-1) description
;; X, Y are position of sonar in mm, TH is bearing in degrees
;;
;;
;; Six forward sonars:
;; # x y th
;;-----
SonarUnit 0 75 105 90
SonarUnit 1 115 80 44
SonarUnit 2 140 30 12
SonarUnit 3 140 -30 -12
SonarUnit 4 115 -80 -44
SonarUnit 5 75 -105 -90

;; These are for the two rear sonars:
;; # x y th
;;-----
SonarUnit 6 -140 -82 -144
SonarUnit 7 -140 82 -216

;; Number of readings to keep in circular buffers
FrontBuffer 20
SideBuffer 40
QuickBuffer 24
```

Appendix C

AmigoWIREFREE Radio Modem Settings

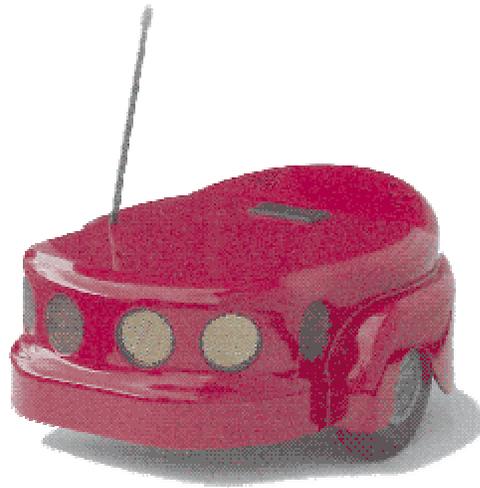
Your AmigoWIREFREE radio modems come pre-configured for use with your AmigoBot. To connect Control Mode software (Saphira and AmigoEYES, for example), all you need to do is attach the host modem to a free serial port on your PC and run the software.

You may examine and alter your AmigoWIREFREE settings, such as to match a new Control Mode baud rate (hostbaud; see Chapter 6). Use Hyperterminal, Minicom, or other simple terminal program. Default settings are DCE, 9,600 baud, 8 bits data, 1 stop bit, No parity. Once connected, all modem control commands begin with "WM". For example, "WMS2" at the host connects the host modem to the robot's modem.

Command	Description
WMBx	Set up the default baud rate. x=1 : 115200 , 2 : 57600 , 3 : 38400 , 4 : 19200 , 5 : 9600.
WMD	Disconnect the radio link established previously.
WMEx	Set up echo and response function. x= 'A' ~ 'P'.
WMFxxxx	Set up the maximum frame length. xxxx must be at most a 4-digit decimal number and ranging from 1 to 1024.
WMIxxxxxx	Set up the group identification code. xxxxxx must be exactly a 6-digit hexadecimal number. The group ID is used to ensure that each connection within the group can be created successfully only if the group ID is the same.
WMJxxx...	Change the identification name to xxx... The length of xxx... cannot exceed 32 letters.
WML	List current setting. The format is as follows:
WMMxxx	Set up my address. xxx must be at most a 3-digit decimal number and ranging from 1 to 255.
WMN	From command mode return to data mode.
WMOxxx...	Set up the partner PN code when creating wireless link. xxx... must be exactly a 32-digit hexadecimal number.
WMPxxx...	Older units have to set up your own PN codes. xxx... must be exactly a 32-digit hexadecimal number. Newer units xxx is a number 1-16; match with pair modem.
WMQx	Query remote setting.
WMRx	Set up the remote output destination. x=P : printer port, x=R : RS-232 port.
WMSxxx	Create a radio link with the partner addressed by xxx. Xxx must be at most a 3-digit decimal number and ranging from 1 to 255. After establishing the link, the async. interface will enter data transmission mode until receiving ESCAPE sequence. The ESCAPE sequence consists of three contiguous ' ' characters and a <CR>. After the reception of ESCAPE sequence, the async. interface will re-enter into command mode.
 followed by <CR> key	From data mode escape to command mode. A delay of 100 ms is needed between the return and any following data input.

Appendix D

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	12V lead-acid
Charge	24.2 watt-hr
Run time	3+ hours
Recharge time (trickle)	8 hrs
Recharge time (fast)	3 hrs

Mobility

Drive wheels	2 solid rubber, with caster balance
Wheel diameter	10 cm
Wheel width	3 cm
Steering	Differential
Gear ratio	19.5:1
Swing radius	33 cm
Turn radius	0 cm
Translate speed max	750 mm/sec
Rotational speed max	300 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) 9,550 ticks per wheel revolution 30 ticks per mm

Electronics

Processor	20 MHz Hitachi H8/2357
Position inputs	4
Sonar inputs	1 x 8 (multiplexed)
Digital I/O	6 digital IO logic ports
A/D	5 @ 0-5 VDC, 12-bit resolution

Digital timer inputs	6 @ 1µsec resolution
Comm port	3 RS-232 serial
FLASH	64 KB µP
	1M external
RAM	16 KB µP

Controls and Ports

Main Power	Robot/accessories power ON/OFF
Charge	System power/battery recharge
RESET	Warm reboot/download
MOTORS/TEST	Motors/download/self-tests
Radio	Power and serial
Speaker	8-ohm
Serial ports	2 x RS232 (Control and System)
	1 TTL (AUX)

Index

- Accessory I/O, 14
- Accessory I/O Ports, 43
- ACTIVMEDIA, ii, 9
- ADSEL, 31
- AmigoBot Mobile Robot, 7
- Amigobot.p, 44
- amigobot-support, 10
- AmigoEYES, 8
- AmigOS, 8, 21
 - amigoscf, 36
 - Client commands, 27
 - Configuration parameters, 36
 - Configuring, 35, 36
 - Download site, 35
 - Downloading, 36
 - Extended packets, 31
 - Installing, 35
 - Programming, 27
 - Restoring parameters, 37
 - Saving parameters, 37
 - Server Information Packets, 23
 - Updating, 35
 - AmigOS commands, 27
 - amigoscf, 36
 - Amigoscf
 - Commands, 37
 - amigosdl, 36
 - AmigoWIREFREE, 14, 45
 - connect, 45
 - parameters, 45
 - Argument types, 27
 - Assembly, 16
 - Audio, 8, 12
 - Autoconfiguration, 27
 - AUX serial packets, 31
 - Auxillary, 43
 - Ayllu, 9
 - Batteries, 40
 - Charge port, 11
 - Charging, 40
 - Battery, 11
 - Low voltage, 11
 - Recharge, 11
 - Recharge time, 12
 - Checksum, 22
 - class, 27
 - Client
 - Commands. *See Client commands*
 - Client commands
 - Argument types, 27
 - Communication rate, 24
 - CONFIG, 32
 - ENCODER, 33
 - General, 24
 - GETAUX, 31
 - SOUND, 31
 - SOUNDTOG, 31
 - Clients, 8
 - Client-Server, 21
 - Client-Server Mode, 9
 - CLOSE, 28
 - Cold Start, 17
 - Comm-port, 36
 - Communication packets, 22. *See Packets*
 - Communications rate, 23
 - Components
 - Battery, 11
 - User supplied, 9
 - CONFIG, 32
 - CONFIGpac, 32
 - Configuration, 15
 - Configuration packets, 32
 - Configuration parameters, 36, 38
 - Control serial, 14
 - Control Serial, 42
 - Controller
 - Specifications, 42
 - Controls, 11
 - Sonar Gain, 13
 - Controls & Ports
 - Charge, 11
 - Main Power, 11
 - Motors/Test, 12
 - Reset, 12
 - Critical parameters, 37
 - Data types, 22
 - DCHEAD, 29
 - DHEAD, 29
 - Digin, 31
 - Dissassembly, 41
 - Drive Lubrication, 40
 - E_STOP, 29
 - Email*
 - amigobot-support, 10
 - amigobot-users, 9
 - saphira-users, 9
 - ENABLE, 28
 - ENCODER, 33
 - ENCODERpac, 33
 - Encoders, 13
 - Errors, 23
 - Extended packets, 31
 - FCC, iii
 - Frequently Asked Questions, 10
 - GETAUX, 31
 - Hardware, 7
 - Hex files, 36
 - I/O, 8, 30
 - Information packets, 23
 - IOPac, 32
 - Konolige, Dr. Kurt, 7
 - LEDs
 - Power, 11
 - System, 12
 - User, 12
 - Maintenance, 40
 - Modes
 - Client-Server, 9
 - System, 12, 35
 - Motion commands, 29
 - Motor power, 43
 - Motors, 13
 - ENABLE, 28
 - Motors/Test, 12
 - Newsgroups*
 - amigobot-users, 9
 - saphira-users, 9
 - OPEN, 28
 - P2OS
 - Configuration Parameters, 38
 - Critical parameters, 37
 - P2OS commands, 37
 - p2oscf
 - Command parameters, 38
 - Packets
 - AUX Serial, 31
 - Checksum, 22
 - CONFIGpac, 32
 - Configuration, 32
 - Data types, 22
 - ENCODERpac, 33
 - Errors, 23
 - Extended, 31
 - IOPac, 32
 - processing, 31
 - Protocols, 22
 - SERAUXpac, 31
 - Server information, 23
 - PAI, 9
 - Physical dimensions, 11
 - Pioneer Mobile Robots, 7
 - Ports
 - Accessory I/O, 14
 - Comm-port, 36
 - Control serial, 14
 - Modem serial, 14
 - System serial, 14
 - Power, 11
 - Auxillary, 43
 - Main, 11
 - Motors, 43
 - PULSE, 28
 - Quick Start, 16
 - Radio Modems, 14
 - Recharge, 11
 - Repairs, 40, 41
 - Authorization, 41
 - Reset, 12
 - Resources, 7, 9
 - Rotation, 29
 - RVEL, 29
 - safety, iii
 - Safety Watchdog, 15
 - Saphira, 9
 - Connection, 17
 - Disconnecting, 18
 - Errors, 19
 - Operation, 18
 - parameters, 44
 - Problems, 19

- Servers, 21
- Startup, 17
- Self-Tests, 8
 - Motors, 20
 - Sonars, 20
 - Wander, 20
- Self-Wander, 20
- Sensors, 7
- SERAUXpac, 31
- Serial, 42
 - Configuration, 42
 - Control Port, 42
 - Internal ports, 43
 - System Port, 42
- Server
 - Information packets, 23
- Server information packets, 23
- Servers, 21
 - ADSEL, 31
 - AmigoBot Operating System, 21
 - Autoconfiguration, 27
 - CLOSE, 28
 - DCHEAD, 29
 - DHEAD, 29
 - Emergency stop, 29
- I/O, 30
- IOREQUEST, 32
- OPEN, 28
- Position integration, 30
- Position registration, 30
- PULSE, 28
- SETO, 29
- SETRA, 29
- SETRV, 29
- shut down, 27
- Sounds, 31
- start up, 27
- SYNC, 27
- VEL, 29
- VEL2, 29
- SETO, 29
- SETRA, 29
- SETRV, 29
- Shut down, 27
- Software, 8
 - Download site, 9
- Sonar Gain, 13
- Sonars, 13
 - Firing rate, 13
 - Range, 13
- SOUND, 31
- Sounds, 12
- SOUNDTOG, 31
- Speaker, 12
- Specifications, 7, 11, 46
- SRI International, 7
- SRI International, ii
- Stalls, 15
- stallval, 15
- stallwait, 15
- Start up, 27
- subclass, 27
- Support, 9
 - amigobot-support, 10
- SYNC, 27
- SYNC0, 27
- SYNC1, 27
- SYNC2, 27
- System mode, 12
- System Mode, 35
- System serial, 14
- System Serial, 42
- Timer, 31
- Translation, 29
- VEL, 29
- VEL2, 29
- watchdog, 15

Warranty & Liabilities

Your AmigoBot is fully warranted against defective parts or assembly for 6 months. Accessories are warranted for 90 days. This warranty 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 AmigoBot 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 AmigoBot Mobile Robots 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 AmigoBot take responsibility for support of any special or custom modification to AmigoBot or its software.



44 Concord Street
Peterborough, NH 03458
(603) 924-9100
(603) 924-2184 fax
<http://www.amigobot.com>