EDUCATOR'S GUIDE TO

ROBOTS

WWW.AMICOBOT.COM
ActivMedia Robotics Educator's Guide to Robots

By Jeanne Dietsch

Thanks to William Kennedy, James Newton and Matthew LeFary for technical assistance with this guide.

Copyright ©2001 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 expressed written consent of ActivMedia Robotics, LLC.

Editor: Elaine Giacomo

Printing History

May 2001: First Edition

The Saphira libraries and Colbert software that accompany the robot and which are available for network download by ActivMedia Robotics customers are licensed from SRI's Artificial Intelligence Center in Menlo Park, CA.

ActivMedia Navigator, ActivMedia Mapper, AmigoSound, ActivMedia WorldLink and other ActivMedia Robotics software are copyright ActivMedia Robotics, LLC.

Duplication, distribution, reverse engineering, or commercial application of any other ActivMedia Robotics software and hardware without the express written consent of the owner, is explicitly forbidden. AmigoBot is a trademark of ActivMedia Robotics, LLC.

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.

ISBN:
# TABLE OF CONTENTS

**CHAPTER 1 **INTRODUCTION

| Uses of ActivMedia Robots | 1 |
| Scope and Sequence | 1 |
| Customizing Lessons by Age and Ability | 2 |
| Equipment Requirements | 3 |
| Educator Support | 3 |
| Robot Placement | 3 |
| Setting up ActivMedia Robots | 4 |
| ActivMedia Robotics Basic Suite Software | 5 |
| Using Simulated Robots | 5 |
| Using ActivMedia Robots over the Internet | 5 |

**CHAPTER 2 **INTRODUCTION TO ROBOTS

| Differences Between Intelligent Robots & Toys | 9 |
| Lesson 1: Robot Navigation and Intelligence | 9 |
| Preparation | 9 |
| Activities | 10 |
| Challenges | 10 |
| Resources | 12 |

**CHAPTER 3 **SENSING & RESPONDING

| The World Goes Digital | 13 |
| From Sound to Bits and Bits to Sounds | 13 |
| The Information Revolution | 13 |
| Digitizing Sense and Motion | 14 |
| Lesson 2: Robot Sensors | 15 |
| Preparation | 15 |
| Activities | 16 |
| Challenges | 17 |
| Resources | 19 |

**CHAPTER 4 **MAPPING & NAVIGATION

| Lesson 3: Mapping & Navigation | 20 |
| Preparation | 20 |
| Activities | 21 |
| Challenges | 21 |

**CHAPTER 5 **NETWORKS & DISTRIBUTED INTELLIGENCE

| Demystifying Networks | 24 |
| Lesson 4: Networks and Distributed Intelligence | 25 |
| Preparation | 25 |
| Activities | 26 |
| Challenges | 26 |
| Resources | 28 |

**CHAPTER 6 **COMMUNICATING ACROSS CULTURES

| Remote Communication: Friends Around the World! | 29 |
| Lesson 5: Communicating with AmigoSounds & WorldLink | 29 |
| Preparation | 30 |
| Activities | 30 |
| Challenges | 32 |
| Resources | 32 |

**CHAPTER 7 **COMMUNICATION AND TEAMWORK

| Communication Has Many Meanings | 33 |
LESSON 6: INVESTIGATING COMMUNICATION MODALITIES & REMOTE TEAMWORK ............................................. 34
PREPARATION.......................................................................................................................... 34
ACTIVITIES............................................................................................................................... 35
CHALLENGES.......................................................................................................................... 35

CHAPTER 8 ROBOTS EXTENDING OUR REACH ......................................................................................... 36
REMOTE SENSING: THE WORLD AT YOUR FINGERTIPS! ........................................................................... 36
LESSON 7: REMOTE HANDLING AND COOPERATIVE TEAMS .......................................................... 36
PREPARATION........................................................................................................................................ 37
ACTIVITIES........................................................................................................................................... 37
CHALLENGES........................................................................................................................................ 38
RESOURCES......................................................................................................................................... 38

CHAPTER 9 INTRO TO ROBOT COMMAND ......................................................................................... 39
ROBOT CONTROLS AND THEIR RELATIONSHIP TO SENSING.............................................................. 39
LESSON 8: INTRODUCTION TO ROBOT COMMAND .............................................................................. 39
PREPARATION........................................................................................................................................ 39
ACTIVITIES........................................................................................................................................... 40
CHALLENGES........................................................................................................................................ 41
RESOURCES......................................................................................................................................... 42

CHAPTER 10 PROGRAMMING ROBOTS .............................................................................................. 43
COLBERT FUNCTIONS....................................................................................................................... 43
COLBERT DEMOS............................................................................................................................... 43
LESSON 9: BEGINNING TO PROGRAM IN THE TRAINER WINDOW .................................................. 46
PREPARATION........................................................................................................................................ 46
ACTIVITIES........................................................................................................................................... 46
CHALLENGES........................................................................................................................................ 47
RESOURCES......................................................................................................................................... 47

CHAPTER 11 RESPONSIVE PROGRAMMING ...................................................................................... 48
TWO APPROACHES TO RESPONSIVE BEHAVIORS.............................................................................. 48
LESSON 10: RESPONSIVE PROGRAMMING .......................................................................................... 50
PREPARATION........................................................................................................................................ 50
ACTIVITIES........................................................................................................................................... 50
CHALLENGES........................................................................................................................................ 51
RESOURCES......................................................................................................................................... 51

CHAPTER 12 PROPRIOCEPTION ........................................................................................................ 52
PROPRIOCEPTION: HOW ROBOTS TAKE CARE OF THEMSELVES..................................................... 52
LESSON 11: PROGRAMMING ROBOTS FOR SAFETY ............................................................................ 56
PREPARATION........................................................................................................................................ 56
ACTIVITIES........................................................................................................................................... 56
CHALLENGES........................................................................................................................................ 57
RESOURCES (for more advanced programming in Colbert)................................................................. 59
APPENDIX A: ROBOTICS HISTORICAL NOTES ................................................................................... 60
APPENDIX B: COLBERT COMMANDS, OPERATIONS AND VARIABLES .............................................. 62
variable types ........................................................................................................................................ 62
Mathematical operations .................................................................................................................... 62
Logical tests ......................................................................................................................................... 63
Robot Motion/drive Commands .......................................................................................................... 63
Program LOGIC, Syntax and Commands .............................................................................................. 64
Robot Status Variables .......................................................................................................................... 65
Partial list of other Colbert Functions .................................................................................................. 67
APPENDIX C: ADDITIONAL ROBOT RESOURCES .............................................................................. 69
INDEX .................................................................................................................................................. 70
Chapter 1

INTRODUCTION

PURPOSE OF THIS GUIDE

ActivMedia Robotics has always catered to the professional or university-level roboticist. With the Educators' Guide to Robots, and our ActivMedia Robotics Basic Suite software, we hope to make our robots accessible to those teaching introductory robotics in junior college, high school or even advanced middle-school courses. Those teaching more advanced levels should check our customer support site: http://robots.activmedia.com.

USES OF ACTIVMEDIA ROBOTS

ActivMedia Robotics Basic Suite is an extraordinarily versatile software tool. It allows nearly any ActivMedia robot to be used for:

Interactive Communications
- Between classrooms
- Among schools
- With sister classrooms
- With parents
- With invalids

Highly Motivating Lessons in
- Computer programming
- Foreign language
- Science
- Math
- Technology literacy
- Social studies

Extracurricular Activities
- Robotics teams
- Science/computer clubs

The critical differences between intelligent mobile robots and toy or model robots are their sensing capabilities. In addition, ActivMedia Robotics robots have sophisticated and wide-ranging software, plug-n-play Internet connection and curricular support in the form of this guide, demos and other materials online.

The more sophisticated the software, the easier the robot is to use. For this reason, we designed ActivMedia Robotics Basic Suite software for use with our robots such as Pioneer 2-DX, Pioneer 2-AT, PeopleBot and in particular AmigoBot. With Basic Suite, ActivMedia robots can be used as successfully by newcomers who have never before seen a robot and by graduate-level roboticists at the world’s leading universities.

SCOPE AND SEQUENCE

Educators' Guide to Robots offers suggestions, projects and ideas for using robots in varied educational settings, under the direction of many different people. Because intelligent mobile robots are so versatile and new options and applications are being developed every month, a comprehensive guide would be impossible. Instead, we have opted to provide sample activities in each venue, for each purpose. We hope that this provides a foundation for long and fruitful use of
robots. We trust you’ll share many new lesson plans, applications and projects with others via the AmigoBot or Pioneer users’ group and other parts of our Support network listed later in this chapter.

This guide begins by discussing where to place robots, based upon the intended use. It explains how the robot can be easily accessed from various classrooms through the school’s internal network connection. It also discusses ways to allow the robot to be used over the Internet by other schools.

The lessons in this guide are grouped according to skill requirements, as shown in Table 1.

Early activities are appropriate for Technology Literacy or Science students ages K-adult. Lessons 3-7 demonstrate robot use for mathematics, computer science, foreign language and communications. Lessons 8-11 teach robotics programming.

Appendices at the end of this guide provide historical information and a Colbert programming reference.

**Table 1. Sequence and Skill Requirements by Lesson**

<table>
<thead>
<tr>
<th>K-Adult (Science &amp; Technology Literacy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lesson 1: Robot Navigation and Intelligence</td>
</tr>
<tr>
<td>Lesson 2: Robot Sensors</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Middle School-Adult (Various subject areas)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lesson 3: Mapping &amp; Navigation (Math)</td>
</tr>
<tr>
<td>Lesson 4: Networks and Distributed Intelligence (Computers)</td>
</tr>
<tr>
<td>Lesson 5: Communicating with AmigoSounds &amp; WorldLink (Foreign Language &amp; Social Studies)</td>
</tr>
<tr>
<td>Lesson 6: Investigating Communication Modalities &amp; Remote Teamwork (Speech/Communications)</td>
</tr>
<tr>
<td>Lesson 7: Remote Handling and Cooperative Teams (Problem Solving)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Advanced Middle School - Adult (Programming)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lesson 8: Introduction to Robot Command</td>
</tr>
<tr>
<td>Lesson 9: Beginning to Program in the Trainer Window</td>
</tr>
<tr>
<td>Lesson 10: Responsive Programming</td>
</tr>
<tr>
<td>Lesson 11: Programming Robots for Safety</td>
</tr>
</tbody>
</table>

**CUSTOMIZING LESSONS BY AGE AND ABILITY**

Because everyone from pre-school children to adults can use simple robots such as AmigoBot, we would be hard put to write a single guide that explains how to customize lessons for every group. We include background information in each chapter to help maintain the interest of the most proficient students. And we provide Challenges at the end of each lesson for beginning, intermediate and advanced students. We leave it in the capable hands of teachers to customize the lessons beyond this point.

We try to provide a smooth interface so that teachers can progress through lessons sequentially, but at some point younger and less able students will need to pause. Later lessons are aimed specifically at students with programming skills, but even these more proficient students must understand that programming robots and programming computers can be quite different. Our appendices provide many useful materials that will help students and teachers in their efforts.
**EQUIPMENT REQUIREMENTS**

To use most lessons in this guide, you will need at least one ActivMedia robot with radio modem, ActivMedia Robotics Basic Suite software and a PC with RS-232 serial port. Having an AmigoBot ePresence system or PTZ Surveillance camera system on each robot is highly desirable. Some lessons can be accomplished with the Basic Suite software Simulator as well as with a real robot. AmigoFingers, the Amigo Delivery Cup or Pioneer and PeopleBot grippers also make lessons more interesting.

**EDUCATOR SUPPORT**

This ActivMedia Robotics’ Educator’s Guide to Robots is part of the support network for educators using our robots. It complements these other support resources:

- **AmigoBot User’s Guide** or the **Pioneer Operations Manual** provides how-to information, and is included with your robot.
- **ActivMedia Software Guide** provides software how-to details, and is included with ActivMedia Robotics Basic Suite software.
- **User’s Groups** are the place to share ideas, ask questions and arrange cooperative activities. To join, email
  
  To: amigo-users-request@activmedia.com; (or pioneer-users-request)
  Subject: subscribe

- **Robotics Newsletters** keeps users up to date on new accessories, software, updates and applications for schools. To join, email
  
  To: amigonews-request@activmedia.com (personal robots news); OR robotsnews-request@activmedia.com (professional robots news)
  Subject: subscribe

- **Support Websites** [www.amigobot.com](http://www.amigobot.com) contains software updates, new product information and FAQ’s. Pioneer and PeopleBot users, see [http://robots.activmedia.com](http://robots.activmedia.com); you will need your ID and password to download licensed software from this site.

- **Free WorldPass Software** for using online ActivMedia Robots from a remote location is available at [www.mobilerobots.com](http://www.mobilerobots.com), our online robot site.

- **AmigoBot Technical Manual** contains college-level programming, applications and interfacing of AmigoBot; technical manuals are included with Pioneers and PeopleBots

- **Tech Support** for repairs and warranty service:
  
  To: amigo-support@activmedia.com; or pioneer-support@activmedia.com
  Subject: help!

**ROBOT PLACEMENT**

Intelligent mobile robots are versatile teaching, communication and demonstration tools for students of all ages. Because computer users can begin to operate them in five minutes, the robots can easily fill the needs of many different students. Some examples include:

- School Technology Lab, by a computer science teacher, for use in computer literacy or computer science studies or as a classroom resource for special projects and demonstrations
• District-wide Central Installation, by a media center or IT department for use as a videoconferencing device for students and teachers, opportunity for parents to monitor their children, or shared robots for programming and use throughout the district

• Classroom Use, by a classroom teacher, for science, computer literacy, programming, foreign language and special projects

• Robotics Teams, by an extra-curricular coach, for robotics contests, club activities and projects.

• School Media Center, by a Media Center Coordinator, for loan to classrooms and support of video-teleconferencing between classrooms.

• Traveling District-wide Resource, by a District Technology Coordinator, for portable demos, weekly instruction or teacher training.

• Daycare or Pre-School, by the Center Director, for video communication with parents, student technology literacy and pre-programming skills.

• Special Needs Room, by the Special Needs staff, for highly responsive and motivating instruction in communication, spatial relations, reading, science, math, technology and other subjects.

SETTING UP ACTIVMEDIA ROBOTS

AMIGOBOT

If this is the first time you're using AmigoBot with these particular PC's, follow the instructions in the AmigoBot User's Guide to set up the robot. If you travel frequently with a single robot, you may prefer to run everything from your laptop. Be sure to order to order the special Laptop Version of AmigoEyes hardware if so. This will include a PCMCIA plug-in framegrabber instead of a desktop framegrabber card.

Wireless radio modems come in matched pairs. If you try to run one robot from another robot's radio modem, it will not connect.

When all hardware is set up and any joysticks calibrated, install Basic Suite software on all computers. Note that you should re-configure the joystick in ActivMedia Navigator even if it is already installed on the computer. Choose Settings: Joystick Calibration to do so. Software installation and start-up should be very simple. If you have any software problems, refer to your ActivMedia Software Guide. For hardware issues, see the AmigoBot User's Guide.

Settings: Safe Speed on gives your robot time to respond to its sonar readings. With safe speed off, the robot is more likely to crash into obstacles.

We recommend always running classroom robots with Settings: Safe Speed on. The reason for this is illuminated in Lesson 2 when students study the speed of sound and the robot's ability to respond in time to sonar readings. Only when robots are running in open spaces such as a gymnasium or for specialized purposes such as a race, should safe speed be turned off.

OTHER ACTIVMEDIA ROBOTS

For other ActivMedia Robots, see the appropriate Operations Manual for hardware set-up. Then follow software set-up instructions for AmigoBot, above.
ACTIVMEDIA ROBOTICS BASIC SUITE SOFTWARE

This guide is based around ActivMedia Robotics Basic Suite software, which includes either three or four modules. Depending upon your robot, your ActivMedia Robotics Basic Suite CD or download will contain:

- Navigator or WorldLink (not both)
- Mapper
- Trainer
- AmigoSounds (AmigoBots only)

Figure 1: ActivMedia Robotics Basic Suite Main Menu (WorldLink edition)

USING SIMULATED ROBOTS

No school can afford a robot for every student, but they can afford school-wide licenses to the ActivMedia Robotics Basic Suite that allows every student program a simulated robot! Your students will have fun playing games with the simulator and gain far more hands-on experience than they could on the actual robot. In addition, you won't need to worry about letting the simulated robot crash into walls! When students can prove to you in the simulator that their programs operate safely and use proprioception to protect the robot, then you can let them connect to a real robot. Seeing the difference between a program that operates perfectly in simulation and not-so-perfectly with a real robot in the real world is an excellent lesson on the difference between working with models and operating in complex, ever-changing reality.

Figure 2: Simulators let students test onscreen without access to a real robot

USING ACTIVMEDIA ROBOTS OVER THE INTERNET

Nearly any ActivMedia robot may be set up for sharing over the Internet with ActivMedia WorldLink software running on the robot's client or host PC. The AmigoBot ePresence system and Pioneer or PeopleBot robots with PTZ Surveillance systems provide video and/or sound over the Internet as well.

Remote users may access the shared robot from any of three software modules: WorldLink, Navigator or WorldPass software. WorldPass is available for free download by non-robot owners at www.mobilerobots.com. They should download the same version as your WorldLink.

If this is the first time you're using online robots over the Internet, launch WorldLink and Edit Users to set up ID's and passwords for appropriate users to connect to the robot. If you're willing to allow anyone on the Internet to use the robot, you may set up an anonymous guest password.

To allow others to share the online robot once ID's and passwords have been created, select Start WorldLink.

Figure 3: Remote users can access a WorldLink-shared robot with WorldLink, Navigator or WorldPass
As shown in Figure 3, the robot to be shared online needs to communicate with a PC running

As in any Internet application, users must know the IP address of their computer in order to allow Internet connections. When a remote user tries to log onto your robot,

a) WorldLink, WorldPass or Navigator software will ask for the IP address of the robot’s host PC.

b) Remote users will also need to enter their ID and password or guest, if you allow anonymous guest access.

c) Remote users must also Take Control before they can have a turn operating the robot. Only one person at a time can control each robot’s movements or make it speak, but many may watch, listen and chat.

If users have trouble following the step-by-step instructions in WorldLink, Navigator or WorldPass, see the ActivMedia Software Guide.

We hope that you will share recommendations and exchange proposals for organizing remote and cooperative learning situations at the AmigoBot Community section of www.amigobot.com or via the pioneer-users group.

ABOUT ACTIVMEDIA ROBOTICS

ActivMedia Robotics has maintained close ties to education since its early days as guest at the Technology Incubator of New Hampshire’s Region 14 District Technology Center. Designed to help bootstrap entrepreneurial companies who might later be able to hire their graduates, the Technology Incubator was a win-win school-to-business partnership. ActivMedia Robotics moved into its own buildings in 1999, but retains its links to education. We are proud to have successfully employed a number of underemployed and difficult-to-place individuals, and assisted these employees in furthering their education.

ActivMedia Robotics also strives to make the latest technologies accessible to students worldwide. Before ActivMedia introduced the Pioneer robot in 1995, intelligent mobile robots were financially out of reach for most universities, much less high schools. Pioneer robots made teaching robotics
affordable to the average university. We hope that AmigoBot will do the same for public school districts.

THE CONTRIBUTORS

Jeanne Dietsch, CEO of ActivMedia, has developed curricula and software for the National PTA, Delta Science, Spectrum-Holobyte and others. Ms. Dietsch co-authored ActivMedia's Science Made Clear with Robots program as well as this guide. Each year she teaches a popular robotics course to area middle school students. She has spoken at Comdex, CES, the annual Williams Syndrome Conference and other meetings on technology use in education.

William Kennedy, PhD, is CTO of ActivMedia Robotics, where he heads R&D and works to make robots more useful and responsive to the people who use them. Dr. Kennedy co-authored O'Reilly & Associates' best selling HTML: The Definitive Guide, now in its fourth printing. He was editor-in-chief of inCider magazine and A+ Publishing. In his leisure time, Dr. Kennedy has taught physics, web design and computer programming to students of The Well School, Mountain Shadows school and New Hampshire Regional Technology Center.

James Newton, Software Engineer, is one of the authors of ActivMedia Robotics Basic Suite software. His mastery of networking made the plug-n-play Internet access of our robots possible. Mr. Newton completed the computer science program at University of Denver and worked as a software engineer for Raytheon before joining ActivMedia Robotics. Mr. Newton reviewed programs and programming instruction in the guide.

Matthew LaFary, Senior Software Engineer at ActivMedia Robotics. He co-authored ActivMedia Robotics Basic Suite, designed underlying ARIA software and leads software design in many other projects. Mr. LaFary earned his BS in computer science from University of North Dakota. He programmed winning robots in the American Association for Artificial Intelligence robot contests two years running. Mr. LaFary contributed to the Colbert appendices in this guide.

Elaine Giacomo, Editor, is a seasoned technologist and instructional editor. She has developed science curricula and testing for Delta Science and worked with PegCo Labs as engineering project coordinator for many years. She was formerly Director of Development at Cedarcrest Support Facility for special needs children.
Chapter 2

Introduction to Robots

Any ActivMedia robot may be used for this introductory activity, but AmigoBot is especially designed to be used with children and adults of any age. The beginning lessons can be as fascinating for senior citizens as for pre-school students.

Figure 4: Children interacting with an AmigoBot

Differences between Intelligent Robots & Toys

What is an intelligent robot, anyway? Many of us have seen people dressed in Star Wars robot costumes, or have observed robot figures operated like remote controlled cars. A remote speaker may let someone standing at a distance talk through the “robot's” body. They are entertaining, but none of these qualify as Intelligent Robots.

Intelligent robots have at least three capabilities: 1) they can track their own movements and conditions, 2) they can sense their environment, and 3) they can make decisions about what to do next, based on the other information. Intelligent robots contain a **microcontroller** or other type of processor that allows them to follow different courses of action under different circumstances.

Lesson 1: Robot Navigation and Intelligence

**Goals**

- To help students discover the differences between an intelligent robot and a toy
- To familiarize students with robot components and capabilities by observing and using the robot

**Who?**

K - adult

**How Long?**

TIME: 25-35 min.

**Where?**

Best conducted in an open space such as an empty hallway outside a computer lab or classroom. Robots should be within 100 feet of the PC's running them if used with AmigoWirefree or most other radio modems.

**Preparation**

**Before Class Begins**

Install and launch ActivMedia Robotics Basic Suite software on each PC you'll be using to connect with robots or robot simulators. Then choose **Navigator** or **WorldLink** from the main Basic Suite menu. Test the connection with each robot by choosing **Robot** or **Simulator** from the **Connection** menu. (For connection troubleshooting, see the AmigoBot User's Guide or Pioneer

**Vocabulary**

- **Autonomous** - runs independently, without help from others
- **Microcontroller** - the robot's “spinal cord” that collects information from its senses, reads if necessary and lines up data to transmit to the PC
- **Navigation** - determining where to go and how to get there safely
- **Teleoperation** - remote control via a wireless link
Operations Manual.) Make sure each robot has Settings: Safe Speed on so obstacle avoidance will function properly.

ORGANIZING
Divide the class into teams. One robot and one PC per team works best, but teams can take turns and some can use simulators instead of real robots.

ACTIVITIES

1) Let students practice teleoperating the robot with arrow keys or joysticks in WorldLink or Navigator. Encourage drivers not to look at the robot as it moves. If robots have cameras, have students drive using the view screen. If no video view is available, ask other team members to watch the robot and tell the driver where to drive. Is teleoperation easy or difficult? Most people that teleoperation looking at the robot’s point of view is fairly difficult, though experienced video game players may not.

2) Can students discover a difference between driving this intelligent robot and driving a toy like a remote-controlled car? If they cannot guess after some practice, suggest that they drive robots slowly toward a wall or toward another robot. What happens? Unlike a remote controlled car, the robot senses and avoids obstacles even when you are teleoperating it. (See Lesson 2 for more on sensors) What makes the robot decide whether to follow your instructions from the joystick or to turn aside to avoid an obstacle? Students may mention sonar, which sense obstacles. The robot also needs to have a small computer called a microcontroller. The microcontroller is the robot’s “spinal cord.” It collects information from the sensors, responds to emergencies, sends information along to the PC “brain” and listens for the PC to tell it what to do.

3) Ask teams to click Wander on the ActivMedia Navigator or ActivMedia WorldLink screen. What does the robot do? The robot moves about randomly, avoiding obstacles. Can students think like robot programmers? Ask students to describe the procedure or steps the robot is taking as it moves about the room. The robot tests its sensing devices (sonar) to see whether an obstacle blocks its way. If not, it may proceed. If so, it will turn aside until it senses an open space ahead. It then moves forward until it senses another obstacle.

4) Next, have students click onscreen somewhere away from the robot icon. What happens on the screen? What does the robot do? The robot plots a path to the point clicked and goes there. When it reaches its goal, it lets you know by saying, “I’m here!” or whatever other sound has been set as its “goal reached” sound. Intelligent mobile robots can navigate autonomously. Is it easier to teleoperate a robot to a place or click on the screen and let it navigate autonomously? Answers may vary under these circumstances. Later experiences will demonstrate that clicking onscreen is definitely easier for more complicated navigation, but either answer is possible now.

5) Ask students to pick up the robots and turn them upside down. Have one team member click onscreen about 100 degrees to the right and away from the robot icon. What happens? The robot’s opposite wheel (left when viewed from the top, right viewed from below) turns until the robot onscreen is heading in the proper direction. Then both wheels turn until the robot reaches its goal.

6) How might the robot know when it reaches its goal? It senses and measures the amount each of its wheels turn.

CHALLENGES
BEGINNER
What use are robotic navigation behaviors? Why would anyone need or want a machine that can find its way autonomously around a building? Robotic vacuum cleaners, lawn mowers and other service robots need to be able to find their way around buildings without assistance. Students may come up with many other practical and entertaining applications for robots.

INTERMEDIATE
Read the History of Intelligent Robots [www.amigobot.com/amigo/history.html] and other links in the Resources section that follows. Then write a brief history of robots. Who invented the word “robot“ and when? Who built some of the first intelligent robots, including some ActivMedia robots' ancestors? How large was Shakey and what could it do? Compare real robots with Star Wars' R2D2 and C3PO. How are real robots less capable? Are there any ways in which they are more capable? According to some scientists, most people have been cyborgs at some point in their life; what are cyborgs and what common machine turns people into them, in some scientists' opinions?

Czech writer Karel Kapeck coined the term robot in his 1920's play “Rossum's Universal Robots”. SRI's Artificial Intelligence Center was among the pioneers building early intelligent robots in the mid 1900's, though mechanical puppets, talking heads and other automata were built as early as the 1300's. One of the earliest intelligent robots, Shakey, was taller than a man and weighed hundreds of pounds. It took Shakey hours to sort blocks with only modest accuracy. Compared with their sci-fi counterparts, contemporary real robots are less able to handle objects, talk with people and sense the world around them than those portrayed in movies. However, they are more capable of communicating data to each other and working in teams than movies portray. Cyborgs are beings that are part human and part machine. The chief criteria for a cyborg is that the machine controls some aspect of the human. Some people consider a person on a bicycle to be a cyborg because riding a bicycle causes a person's heart to pump faster, which affects other bodily functions.

ADVANCED
In WorldLink or Navigator, Connect to one of the robot Simulators instead of a real robot. Then select Map/Load Map and select the office.wld map file from the right side of the screen. Map files provide sensory input for the simulated robot to use as it navigates.

Once the map appears, you will see how the sonar are affect by the walls in the map. Click the - button to see more of the map. Click on various parts of the map and watch the robot plot a path to each. Try to find a place on the map to which it cannot calculate the path. Can you fool the robot Simulator? Probably not. Does the robot always find the shortest path? Nearly every time. What steps do you think the robot might follow to calculate such a path between any two points, avoiding walls and other mapped obstacles as it does so? Even calculating a path between two points on a map requires some intensive mathematics, much less planning paths around obstacles.

Try a simpler problem. How would you tell a robot to move from point x to point y in an empty room, just using a simple straight line if the robot is already heading in the right direction? If the robot's heading were already correct, you would only need to calculate the distance between the two points using $d^2 = (y_2-y_1)^2 + (x_2-x_1)^2$. Since the map coordinates are in millimeters, this distance would also be in millimeters. To determine how far to move the robot's wheels forward, you would need to know how many millimeters the robot travels in one turn (500 ticks) of its wheel by measuring the wheel's circumference in millimeters. Dividing the total number of mm distance by the mm circumference would yield the number of times the wheel must rotate in order to reach the goal. Multiplying that number by 500 would yield the number of ticks the encoder must count. Thus, even calculating a simple behavior like moving forward a certain distance in a straight line requires a fair amount of work!
\[ d = \sqrt{(y_2-y_1)^2+(x_2-x_1)^2} \]

number of ticks to move forward = \( \frac{d}{\text{wheel circumference}} \times 500 \text{ ticks per revolution} \)

RESOURCES


"History of Robotics," www.ukrobot.co.uk/html/education/data/history.htm

Chapter 3

SENSING & RESPONDING

For many people, digital information - i.e., computer programs and data - is something that sits inside a computer and affects only what is seen on a screen. Robots dramatically demonstrate that digital information can move out of the box and into the real world, recording measures of the surroundings and responding with actions.

Sensing and manipulating one’s world through the medium of machines is nothing new. Ever since they invented the lever and the sundial, humans have extended their grasp and their sensitivity to changing phenomena around them.

Figure 6: Hobbyists at the annual Firefighter’s contest inspect an AmigoBot

THE WORLD GOES DIGITAL

Technologies of the Industrial Revolution allowed people to create more clothing, household goods, and other items in far less time. After Henry Ford introduced the concept of mass production in the early 1900’s, manufacturing became even faster and cheaper. By the end of the 20th century, the Information Revolution had taken society to yet another level of productivity. Many of the physical components of manufacturing, transportation, and communication are now being transformed into data. Because transporting data can cost a thousand times less and occur a thousand times faster than moving materials, business owners are racing to convert every possible facet of their companies into data. This digital transformation is every bit as monumental as the Industrial Revolution.

FROM SOUND TO BITS AND BITS TO SOUNDS

Music is probably the example most familiar to students. Before the Information Revolution, people bought records or tapes or CD’s - physical objects they carried home from a store. Now music is information. When artists record music, thin plates inside a microphone vibrate from the sound waves. These vibrations are used to create analog electronic waveforms. Electronic digitizers such as Sound Blaster cards sample the sound waves at various points along its curves and convert the current values into numbers. The more frequently the sound is sampled, the closer the digitized (discontinuous) version will be to the analog (continuous) version. Once the digital sounds are collected, edited and electronically altered to the artists’ satisfaction, they are then stored in a final version that is made available to the public.

The storage medium can be whatever the listener wants it to be: CD, memory stick, computer hard drive, or other device. That is the physical storage medium that we have shipped to us or pick up at the store. But the artistry, the sound and the delight of the music or the music video is in the information - the bits and bytes - that we download and/or play. Once we have stored the information, it must then be converted back into sound. A CD player, for instance, translates digital signals into electronic pulses that are then magnified in the amplifier, which sends electrical current to vibrate the fiber membranes of the various speakers or headphones attached to it. These vibrations send out sound waves that then reach our ears.

THE INFORMATION REVOLUTION

The music industry is transitioning from tapes, CD’s and physical media to downloaded information. Likewise, other companies are in the processing of transforming their physical products into information.
Regional newspapers download their day's news to a local printer who then delivers to customers nearby. Some day, blue jean designers may download their latest designs to a local retailer whose computer-aided-manufacturing equipment cuts and sews a pair with your personal size requirements custom fit for you to pick up next morning.

Digitizing Sense and Motion

What does all this have to do with robots? Well, robots just happen to be the physical "something" that will be involved in many industrial, commercial, healthcare, military and other digitized operations. Robots transform physical characteristics of the environment into digital sensory data that can be sent to people or computers. Robots also turn digital data received from computers or humans into physical motion.

Almost all ActivMedia robots have at least two main sensor systems for discovering environmental characteristics and digitizing sensory information. They also have at least two effectors for turning information into action. Most ActivMedia robots' primary sensors are their eight, sixteen or thirty-two sonar and their two motor encoders. Their main effectors are their two wheel motors and their sound system.

Robot sonar collect information about the walls, creatures and objects in its environment. Like bats, they emit a sound, the clicking you hear when you connect to an ActivMedia robot. This sound moves at about 1080 ft. per second from the sonar to any object or wall that lies ahead of it. If it meets an obstruction, it bounces back. If it rebounds at an angle that the sonar can detect, the sonar electronics calculate the distance and send that information to the robot’s microcontroller. The range of the sonar is about 20 feet. If it doesn’t hear any returning click within the period of time required to travel that distance and back, it records that reading as zero. The reading is a digital record of the distance between the sonar and the robot at that moment in time.

All ActivMedia robots begin numbering sonar with 0. Beginning numbering with 0 instead of 1 is a convention frequently followed by programmers. Some robots may have as many as 32-sonar, numbered from 0 to 31. The sonar layout for AmigoBot is shown in Figure 1. For other robots' sonar layouts, see their operations manuals.

Most ActivMedia Robotics robots also have motor encoder on each drive wheel. AmigoBots and Pioneer 2-DX's have 500-tick motor encoders. That means that every time a wheel makes one complete forward revolution, the encoder count increases by 500. (For other robots' encoder specifications, see their operations manuals.) The motor electronics send the encoder count information to the microcontroller as well. By comparing increases, decreases and differences between the counts of both wheels, AmigoBot’s navigation software determines where the robot is headed in relationship to its original position, and how far it has traveled.

Other sensors include electronics that sense battery charge and motor stall. Some ActivMedia robots also have cameras that can act as color sensors, infrared (IR) sensors for docking and re-localization, lasers for precise navigation, thermometers and smoke detectors for hazard sensing,
AmigoBot motors themselves are effectors; they move in response to commands from the microcontroller. When the software underlying Navigator, WorldLink or WorldPass is told to move the robot to a particular location, it tells each motor to start moving. When the encoders report that the motors have spun the required distance, the navigation software issues the next command: to stop, to change direction, or to move at a different speed.

All AmigoBots also have audio systems that allow them to play recorded sound upon command from the microcontroller. Some of these sounds are played by AmigoBot software to communicate the status of parts of the robot - for instance, when the robot is connected or when its battery is running low. Other sounds are available by pulling down the menu in Navigator or WorldLink.

Some ActivMedia robots may also have other effectors such as grippers to pick up objects.

LESSON 2: ROBOT SENSORS

GOALS

• To understand how robots and other machines sense the world
• To learn how various senses complement each other in machines, animals and people

WHO?

K - adult

HOW LONG?

TIME: 25-35 min.

WHERE?

Best conducted in an open space such as a classroom with desks pushed against the wall or an empty hallway outside a classroom or lab. Robots should be within 100 feet of the PC's running them if used with AmigoWirefree or most other radio modems.

PREPARATION

BEFORE CLASS BEGINS

Launch Navigator or WorldLink software on each PC and test the connection with each robot. Make sure each robot has Settings: Safe Speed on. Then disconnect all robots and turn them off.

ORGANIZING

Divide the class into teams. One robot and one PC per team works best, but teams can take turns and some can use simulators instead of real robots.

ACTIVITIES

1) Set a robot on the floor in plain view and turn it on. Don’t Connect from the PC to the robot yet. Instead have a student push twice on the black Motors/Test button on top of the robot. The robot
will start to move by turning in each direction. Press the Motors/Test button one more time and listen carefully. What noise can students hear? Clicking noise. Can anyone identify which part of the robot is making the clicking? One of the round gold sonar circles. Does anyone know why it clicks?

DO NOT SET A ROBOT ON A DESK, TABLE OR OTHER RAISED SURFACE,
If you press the Motors/Test button, click on the ActivMedia Navigator display, operate the joystick or do anything else that causes the robot to move, it may drive off the edge!

2) Remind students that they saw the robot avoid walls and other obstructions in Lesson 1. How did the robot sense walls? Sonar sensors, the eight round disks located around the robot, click in order to measure the distance between themselves and the nearest obstruction. What animal uses a similar technique? A bat.

Figure 8: Blue rays show the robot's sonar readings

3) Bats and many mobile robots use echolocation to navigate. To demonstrate how sonar echolocation works, set robots on the floor as far away from each other and objects as possible. Have team members turn on the robots, while others watch the host PC displays, and Connect from Navigator or WorldLink. Watch the sonar map that appears. The red icon in the middle of the sonar map symbolizes the robot and the blue rays on the screen indicate the robot’s sonar readings. Have some team members surround the robot, approaching and then backing away, as others remain watching the display. What happens on screen? The blue sonar cones grow and shrink depending upon how distant or close people are to the robot. Measure the farthest and closest distances the robot can sense. What is the functional range of the sonar? Typically, AmigoBot sonar can measure from 15 cm (6 inches) to 6 m (20 feet).

4) How do sonar and bat's sensing work? Students probably know that bats use sound to sense their prey, but may not understand that sonar measure the time of flight between the emission of the sound and its receipt back. Ask if anyone knows the speed of sound, or have students find it in an almanac. How long would sound take to travel 20 feet and back? Six inches and back? Why might it be difficult for sonar to ever be able to measure ranges less than six inches? In air at sea level, sound averages 1,088 feet per second, so it would travel 40 ft in less than .04 second. Sound can travel 12 inches in less than .001 second. Speeds faster than this would be very difficult to measure so sound wave rebound will probably never be the best way to measure small distances.

5) Compare the video image in Navigator with the sonar map beside it. What is different about the image created by the robot’s sonar and the picture created by its camera? (If your robots do not have cameras, ask students to compare their own vision with a robot's sonar sensing.) HINT: Can you see objects behind you? What is the point of view? With their thousands of rods and cones or pixels, human vision or a robot with camera gains more detailed information. Sensors like sonar only generate a single number - time of flight. On the other hand, robots can sense in many directions at once so they know more about their position in space.

6) If possible, ask some students to draw the shades and turn off the lights in the room as others continue to watch the screen. What happens to the robot’s camera view? What happens to the sonar map? The visual view, of course, goes dark, but the sonar display remains. Why might bats find sonar
Bats are nocturnal hunters so their “sonar” allows them to sense prey on even moonless nights.

CHALLENGES

BEGINNER

Blind people sometimes use canes to help them navigate, tapping the cane in front of them as they walk. Using what you know about robot sensing, guess how cane tapping helps make up for a lack of vision. The sound of the tap would echo more quickly if a blind person were heading toward a wall or other obstacle. Blind people's hearing becomes more acute to compensate for their lack of sight, so they can tell by the difference in sound that an obstacle is near. What advantage do blind people using a cane have over people relying on sight? They can avoid obstacles just as well in the dark as in daylight.

Test your hearing. With eyes closed, tap a cane or similar item on a bare floor like a gymnasium. Using the cane, walk toward a wall, beside a wall and away from a wall. How sensitive is your sense of hearing? Can you notice the difference in sounds? Can you walk toward the wall and stop a foot away without opening your eyes?

INTERMEDIATE

All sensing systems have strengths and weaknesses. By understanding a sensor's weakness, we can plan for situations in which it might be mislead. For example, artists painting pictures often use perspective, an optical illusion that fools human vision into perceiving a flat surface as three-dimensional. Likewise, sonar and motor encoders can be fooled by certain stimuli into misperceiving. Think about how sonar or motor encoders work. What might trick them into misperceiving? Let students compare sonar readings from the following objects, held at various angles to the robot's sonar:

- A glass, mirror or hard, smooth plastic sheet
- A large book
- A person's leg
- A very thin chair leg
- A thick chunk of foam rubber

Which items are good sound reflectors? Which are not? What are the other limitations of sonar?

Sonar emit clicks and measure the time required for them to rebound. But what if only a tiny portion of the sound waves meets with a narrow chair leg? What if the sounds rebound off a slanted wall and the rebound never reaches the robot? Or what if the wall is covered with thick, acoustically dead material, like foam rubber? What if one robot hears another robot’s sonar clicking and mistakes it for its own? While sonar are the best reasonably-priced range-finding system available, they have limitations. More accurate sensor systems, like laser range finders, measure the time of flight of light. They improve their accuracy by making hundreds of measurements around them, instead of eight. But lasers cost hundreds of times more than sonar and are quite heavy, so they run only on larger commercial robots like ActivMedia PeopleBot and Pioneer robots.

ADVANCED

a) Motor encoders are another type of sensor included in all ActivMedia robots. Motor
ActivMedia Robotics Educators' Guide to Robots

Lesson 2: Robot Sensors

Encoders measure the turning of the wheel shafts. The motor encoders on AmigoBots and Pioneer 2-DX's are very precise; they measure 500 ticks for every rotation.

b) Launch Trainer from the main Basic Suite menu and connect to a robot. Tell the robot to turn 90 degrees by typing the following on the Trainer display:

```
Turn To (deg): 90
```

Press Enter on the keyboard and watch the robot's wheels. What action does the robot take in order to turn? It rotates one wheel and leaves the others stationary. If a full rotation is 500 ticks, how many ticks did the robot's encoders count before the program stopped? 125 ticks. How can you discover how many inches does the robot move forward for each 500 ticks the motor takes? You might type 500 after Forward on the Trainer display, measure how far the robot goes. Another means, possibly more accurate, would be to measure the circumference of the wheel, which is 12.75 inches. Then divide 12.75 inches by 500 ticks to find out how many ticks the motor takes to move the robot forward 1 inch.

Fill in the table to show how many ticks each of the robot's motors should turn in order to follow the path shown. The last arrow is meant to backtrack over the preceding path.

![Diagram](image)

<table>
<thead>
<tr>
<th>Step</th>
<th>Left Motor</th>
<th>Right Motor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Forward 4 ft</td>
<td>1,882</td>
<td>1,882</td>
</tr>
<tr>
<td>2 Turn 80° right</td>
<td>111</td>
<td>0</td>
</tr>
<tr>
<td>3 Forward 2 ft</td>
<td>941</td>
<td>941</td>
</tr>
<tr>
<td>4 Turn 50° left</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>5 Forward 5 ft</td>
<td>2,352</td>
<td>2,352</td>
</tr>
<tr>
<td>6 Backward 5 ft</td>
<td>-2,352</td>
<td>-2,352</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) In using encoder counts to determine how far the robot has traveled, what problems might a roboticist encounter?

If the wheels are running on a slick or sandy surface, the wheels will turn but the robot will not move as much as it would on a flat floor with good traction. Similar error is introduced when the robot climbs a doorsill. Its software doesn't know that part of its motion is up and down rather than forward. Therefore,
the software will be fooled into misjudging the robot’s position. (AmigoBot has very sophisticated software techniques discussed later that counteract this problem.)

RESOURCES

Descriptions of other types of robot sensing devices such as laser ranger-finders and GPS systems can be found at ActivMedia Robot Accessories: http://www.activrobots.com/ACCESSORIES
Chapter 4

MAPPING & NAVIGATION

Industrial robotic arms perform repetitive tasks like painting cars or inserting screws. An expert operator trains the robot's arm in the path that it needs to follow in order to paint its portion of the car or insert the screws. Once trained, the arm can follow that path repeatedly until it's stopped or requires maintenance.

Today's mobile robots are learning to find their own way around, without being driven or specifically instructed path by path where to move. With ActivMedia Robotics Basic Suite software, a robot can find its way around a map, from classroom to classroom. Robots that can navigate spaces on their own are able to operate in changing environments where people and objects move about. People can operate more than one robot at a time when the robot is intelligent enough to find its own path. Under adverse conditions, the robot may be able to find its way more quickly than a human teleoperating the robot by sight.

ActivMedia robots determine their location by comparing their sensor data with that on their map. Using probability formulae, they calculate the most likely actual location. Most ActivMedia robots use sonar to determine their sensed position. Then they determine their mapped position by calculating the path as recorded by motor encoders since last localizing. The robot then compares its mapped location with its calculated location using a probability formula to determine its most likely actual location. Once calculated, this becomes the new mapped position. The software ignores temporary readings from people walking by or from other moving objects when it localizes, which improves its accuracy.

LESSON 3: MAPPING & NAVIGATION

GOALS
• To build a map that a robot can use to navigate
• Test delivery by robot

WHO?
Anyone ages pre-school - adult

HOW LONG?
TIME: 35-45 min.

WHERE?
Choose an empty hallway outside a classroom or lab. No part of the area should be more than 100 feet from the PC's running the robots unless you have long-range modems or onboard PC. It's best to start with a simple space with one or two doorways and one corridor intersection or turn. Note: stay away from walls that are glass at floor level or acoustically dead surfaces like heavily padded cubicle walls.
Your map should focus on the hallway as shown in Figure 12. Doors that will be closed, that the robot should not enter, should show no entry on the map. Doors that the robot may enter should appear open. Have students measure each wall of the hallway and each entryway.

**PREPARATION**

**BEFORE CLASS BEGINS**

a) Gather 2-4 tape measures per team, preferably metric for more accuracy.

b) Launch Mapper software from the main ActivMedia Robotics Basic Suite menu and familiarize yourself with its operation. (See ActivMedia Software Guide for details.)

c) For younger children, draw an outline map of the space the class will be measuring so they only need to add dimensions. Make copies for each team plus a few extras.

d) Launch Mapper on each PC and test the connection with each robot. Make sure each robot has **Settings: Safe Speed** on, then disconnect all robots and turn them off.

**ORGANIZING**

Divide the class into teams, one team for each computer with Mapper.

**ACTIVITIES**

1) Explain that students will be creating a map for the robot to navigate. Here are a few mapping tips:

<table>
<thead>
<tr>
<th>Table 2: Tips for building useful maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Since this map is for the robot, from what point on the wall should students measure? What will the robot sense? In some cases, it won't matter, but if the wall has baseboard trim, measure at the height of the robots’ sonar.</td>
</tr>
<tr>
<td>b) The map's wall lengths and doorway widths must be very accurate, within a quarter inch or half cm, to work properly.</td>
</tr>
<tr>
<td>c) Do not include objects sitting in the hall unless they are permanently fixed in place. Likewise, don’t try to include desks and chairs in the rooms. The robot will ignore moving objects and less important features of a room.</td>
</tr>
<tr>
<td>d) While the width of walls and doorways needs to be measured carefully, small differences in depth, such as that shown in the closed door in Figure 12, are not important. The entire wall, including the closed doorway, may be treated as a single length.</td>
</tr>
</tbody>
</table>

**MATERIALS**

- 1-4 AmigoBots or other ActivMedia robots
- ActivMedia Navigator or WorldLink software
- 1 PC for each robot
- Tape measures, preferably metric, 2-4 per team
- Optional map outline w/o dimensions
- Optional joystick
- Optional: duct tape to mark robots’ Home position
2) Assign each team a section of the space to be measured. Have them draw a plan of the entire space and indicate measurements on their own section. Rotate groups so that each group measures at least two sections and each section is measured at least twice. Remind students that measurements must be accurate to a half cm or quarter inch. Compare measurements and re-measure any areas of disagreement.

3) In Mapper have each team draw a map of the space. (For details on using Mapper, see the ActivMedia Software Guide.) Label rooms with goal markers. They will then appear as goals in the menu of Navigator or WorldLink. Have each team set a different Home position from which they will start their robot.

4) Test the maps to see which team's works best. Set the robot in its Home position. Make sure the robot is facing exactly the same direction as the robot in the map. Then click on a map location away from the Home position. Choose a goal about equidistant between all the robots' Home locations. If you have AmigoFingers on AmigoBot or a gripper on a Pioneer or PeopleBot, give each team a different colored block and see who can deliver it to the chosen location first. Or if you have a Delivery Cup on AmigoBot, write a message such as “I won!” and place it in the cup. Have each team click on the point at the same time and see whose robot arrives with the block or message first.

When testing a map, make certain the real robot is oriented in exactly the same direction as shown on the map.

5) Watch the robot decide on a path, plotting the way to its goal. If the real robot encounters an unmapped obstacle along the way, the planned route will change as the robot reacts to its environment. When it reaches its goal, AmigoBot will tell you so. Other robots will typically just stop, but the Navigator or WorldLink status window will indicate that the goal was achieved.

6) Now the true test - click again on the robot's Home and watch it return. If it arrives safely, the map passes the test!

7) If students' maps don't work, follow the steps in Table 3:

| Table 3: Troubleshooting Maps |

If your robot frequently gets lost in your map

a) Compare the problem map with maps that work. How is it different?

b) Make certain the robot is heading in the correct direction from the Home base. Compare its sonar readings with the walls showing in the map before trying again. If they don't seem quite right, re-position the robot to fit the map more accurately.

c) Is part of your map based on a large non-acoustic item such as thick foam rubber or smooth glass? These surfaces do not reflect sonar well, so the robot will not sense them. Erase them from your map. (But be careful - the robot might bump into them)

d) Although the robot can account for some variation in its environment, if there are many people milling about, the robot will become confused. Move people out of the way, or wait until a quieter moment and try again.

e) If you want, you can tell your robot where it actually is on the map. Choose Navigation/Relocate from the Navigator or WorldLink menu. Then click the robot's actual location on the map. Move the mouse and click once more to set the robot's mapped orientation to match its actual position.
8) Once the map is working, save it so you can use it to send your robot on errands. If school regulations allow, mark the robot's Home spot and orientation on the floor with duct tape. This will allow you to accurately place your robot for future use with the map. If not allowed, note precise measurements and angle of your robot's home position so you can place it there again as needed.

**CHALLENGES**

**BEGINNER**

In Navigator or WorldLink, **Connect to the Simulator instead of a robot.** In the **File menu**, students should select **Load Map** and choose the map created by their team. Ask them to drive the simulated robot around the map. How does it perform? What is the difference between operating a robot in the simulator and in the real world?

In the simulator, there are no people, no chair legs or other unmapped objects to interfere with the robot’s navigational planning. The robot will nearly always be able to reach its goal easily in simulation. This is why real world testing is so important to both objects and ideas. What works in theory or in a model, may not work in practice because reality is much more complex.

**INTERMEDIATE**

If a robot docking station is available, set it up somewhere in the mapped space where it can be left for long periods. (If a docking station is not available, have intermediate students do the Beginner or Advanced activities.) In Mapper, load the map made during class. Erase the existing Home position of the robot and make the docking station the new Home position.

Set your physical robot at a distant corner of the map ped space. From Navigator or WorldLink, click on the Navigation menu and choose Relocate. Move the robot's icon to its current location on the map and click. Be sure to orient the robot in the correct direction before clicking again. Is the robot icon placed correctly on the map? If not, repeat the Relocate process.

When the robot is oriented accurately, click Home in Navigator or WorldLink. Your robot should head off to be charged. Later on, when you learn to program, you will be able to write a program that sends the robot **Home** whenever it senses that its battery is low!

**ADVANCED**

Create a school robot demo using Navigator and a map. Extend your map to include a 150 foot radius in all directions from a PC's radio modem in an area that requires frequent deliveries or for demos. The route between a classroom and the library or a classroom and the office might be a good choice. (If your robot travels beyond radio modem range, it will say "losing communications" or whatever system message has been custom inserted to indicate communication loss. Then it will stop.) If you have access to a laptop that can be carried along with the robot, you can map your entire school. Try sending your robot from room to room on the map and refine until the robot can travel easily to most or all destinations.

Once the map is perfected, use it for school demos, parades, deliveries, pep rallies or other events. You can attach a bicycle flag or banner to the robot, have it deliver a bouquet of flowers, use it to monitor hallways, etc. Large rooms such as gymnasiums require special retrofitting because sonar measure only about twenty feet out. If using the robot in a gymnasium that will be filled with people, do include the block of chairs as a map object even though it's not permanent. If the audience will be sitting in bleachers along the sides, students may want to position canisters or other sonar-detectable markers as landmarks for the robot to use during its performances.
Chapter 5  

**Networks & Distributed Intelligence**

Because communications, networking and distributed intelligence are becoming increasingly integral to workplace, home and education, students need to understand enough about these technologies to use them capably and comfortably. Robots like the AmigoBot or Pioneer provide an easy way to demonstrate, teach and use networking and distributed intelligence.

**Demystifying Networks**

Demystifying communications and networking is critical to technological literacy in the Information Age. Telecommunications companies like to say that they connect computers to an “Internet cloud.” Many people feel as if they’re in a cloud when trying to understand how networks function, but the basic concepts are quite simple. Most of the lesson that follows is for robots with WorldLink, since its specialized software allows robots to be controlled online. However, the first few activities are designed to help even those without an online robot to begin to understand communications, networking and remote use.

One of the differences between machine intelligence and human is that we can move machine’s “brains” and “spinal cords” to wherever we find them most useful. In the case of ActivMedia robots, we’ve put the microcontroller - the spinal cord that reads sensory input and controls motor output - on the robot. The brains may be off board, in the computer software on the desktop PC, or on very sophisticated robots, the brains may be in a PC inside the robot. Using an existing desktop PC to run the more complex software allows a sophisticated robot to be built less expensively.

Another difference between humans and robots is that all human internal connections are hard-wired with tangible nerves while robots can use radio waves to connect parts of their nervous systems. Just as sonar emit sounds and detect them, radio modems also emit radio signals and listen for them. Robots
radio modems listen for signals of specific frequencies (in the US, between 902 and 928 Megahertz (MHz)). Radio waves in this range are then translated by the modem into numerical codes sent through a serial port. These codes tell the robot’s microcontroller software what to do, or tell the PC’s AmigoBot software what the robot has sensed.

But communication between robot and host PC is just the beginning for a machine brain. What’s most different about a robot’s brain is that it can communicate with any other computers or robots-connected-to it; this happens over the Internet. But how does the Internet connect?

Often, in a school or business, a Local Area Network or LAN connects all the computers with a wire that looks very similar to (and sometimes may actually be) telephone wire. These wires accomplish the same thing as the radio modems sending radio waves through the air, but they send electronic signals through the wire instead. Computers and robots connected by LAN talk to each other directly, but to communicate outside the local area usually involves another network, the public Internet.

If a LAN is connected to the Internet, one of the computers on the LAN functions as an Internet server, controlling access to the Internet. This computer then connects to the computer of the Internet Service Provider (ISP). Whether a LAN server or a home PC, the ISP connection might be made via a cable TV line, a telephone line or satellite feed. Of course, the ISP probably connects to another computer, which connects to another; some Internet connections can involve dozens of computers. With all these computers connecting, how do they know where to send the data?

When radio modems are only looking for a specific frequency, they assume that anything in that frequency range is intended for them. For instance, a radio modem looking for frequencies between 902 and 928 MHz will listen to everything it senses in that band. But on a network, all the information intended for all the computers travels together. Every device on a network – whether it's a PC or a robot or a printer or a scanner – must have a number that's used to identify it. The most common means of identifying devices is to issue them an IP address. The advantage of this method is that the device needs only a single address, whether it's being accessed over a LAN or over the Internet.¹

That’s why, when connecting to WorldLink on a robot host PC from WorldPass, Navigator or WorldLink on a remote PC, you need to type an IP address. This address tells the network which computer hosts the robot.

**LESSON 4: NETWORKS AND DISTRIBUTED INTELLIGENCE**

**GOAL**

To demystify some of the more common technical aspects of distributed intelligence, networking and wireless communications used by robots, computers and other devices

¹ In fact, every Internet URL you type is linked to an IP address. The master list of URL/IP addresses is kept on each network’s DNS (Domain Name Server) and is updated daily by each Internet server.
WHO?
Middle school - adult

HOW LONG?
TIME: 25-35 minutes

WHERE?
Best conducted in an open space such as a gymnasium a wide empty hallway outside a computer lab or classroom. Robots should be within 100 feet of the portable or desktop PC’s running them.

PREPARATION
Launch ActivMedia Navigator or WorldLink software on each PC and test the connection with each robot.

If you have WorldLink and have not yet set up user profiles for remote users, see “Using ActivMedia Robots over the Internet”, page 5, or the ActivMedia Software Guide for set-up instructions. If you have run robots with WorldLink before, just test the connection with one or more remote PC's using your robot.

If you do not have WorldLink, download the latest version of free WorldPass software from www.mobilerobots.com (1 minute or less, even on a slow connection). Choose the version that matches the robot’s. Launch WorldPass and Connect with an online robot such as that at WorldLink Serverwebpion.mobilerobots.com (See "Using ActivMedia Robots over the Internet," page 5.)

Make sure each robot has Settings: Safe Speed on. Before class begins, leave software running, but Disconnect from the robots and turn them off.

ORGANIZING
Divide the class into 2-4 teams. One robot and one PC per team works best, but teams can take turns and some can use simulators instead of real robots.

ACTIVITIES
1) Set robots on the floor. Have a member of each team tum on the robot. Ask whether robots are connected to their host PC’s. If students are uncertain, have them select Connect: To Robot in WorldLink or Navigator and listen to the robot say, “Connected.” Then have them choose Disconnect from the Connect menu.

2) Once they're certain that the robots are not connected to the PC’s, ask a team member to press the black Motors/Test button twice and then step away from the robot. What happens? The robot turns clockwise, then counterclockwise, and stops. After the robot stops moving, ask team members to observe the robot as one team member presses the Motors/Test button again. Observe the results. Robot will test each sonar sequentially each time the button is pressed. Continue pressing until all sonar have clicked and the robot begins the final step of its self-test: wandering on its own. Ask students to test the robot's obstacle avoidance by stepping into the robot's path (but not too close -- sonar need at least 15 cm (six inches) away from an obstacle to stop).

3) Ask students what they think the robot was doing during the self-test. It was testing each motor and each sonar. Students could see wheels move and hear each sonar click. Then the robot went into random wander mode, moving about and talking to itself and avoiding obstacles. How can the robot perform such a variety of functions by pressing one button if it's not connected to the PC? The robot has a computer chip with software programs inside on its microcontroller. The microcontroller acts like the spinal cord of the robot. It can “ping” the sonar and control the motors, just as an animal's spinal cord can operate muscles without input from the brain. What is the robot not able to
do without connecting to the PC? It cannot follow a path or find its way around a map. It cannot be teleoperated, transmit audio or display its video and sonar readings onscreen.

4) Have each team connect with its robot. What happens onscreen? The sonar readings appear and, with the proper hardware, a video image appears. Does anyone know how the robot connects? What devices are connecting and what do they do? The narrow antenna sticking up from inside the robot is the radio modem to transmit computer signals. The wider, shorter antenna atop the robot is the audio/video transmitter. As its name suggests, it sends the sound and image signals to the receiver attached to the host PC, while the radio modem sends signals to its partner on the PC. Have students look at where the radio modem connects to the PC. That is the serial port, for data. Where is the audio/video receiver connected to the PC? Those connections are the audio and video input ports.

5) With more than one robot, have students disconnect their robots and turn off all but one. What happens if they try to connect that one robot with another robot's host PC? They can't connect. The radio modem pairs are matched to particular frequencies. They only listen to other modems at exactly the same frequency.

6) The Internet is not like radio modems. It can connect to many different devices because it recognizes the identifying number or IP address of each computer or printer or robot linked to it. If WorldLink is available, launch it, connect to a robot and make the robot available over the Internet. Choose Settings from the WIN Start menu. Then select Network. Highlight a TCP/IP setting and click Properties and then IP Address to see the IP Address of your WorldLink host. Once you know the IP address of the WorldLink server, you can connect to and run the same robot from many remote computers.

If WorldLink is not available to use with your robot, students can run ActivMedia's online robot by launching WorldPass software on a computer connected to the Internet. In WorldPass, Connect and choose the default WorldLink server to see our Robot Game Room and take turns running our online robot.

    webpion.mobilerobots.com.
    ✓ Guest Login

7) What is different about running a robot online versus running it directly without the Internet? Response time is slower because commands to the robot must travel through all the Internet connection to reach the WorldLink server.

8) In addition to robot commands, what other types of data can be passed over the Internet with WorldLink and WorldPass? Picture information is transmitted from the robot's camera. If you robot has a camera, have students take Snapshots of what the robot sees. In addition, text files can be exchanged by selecting Chat. If speakers are attached to the PC they're using, students should also be able to hear any sounds near an AmigoBot ePresence or other robot with microphone. All these types of information transfer are useful for mining, exploration, equipment maintenance, entertainment and many other applications.
CHALLENGES

BEGINNER
Like every technology, networks have beneficial and potentially harmful applications. (See Resources, below, for sites that discuss some of technologies benefits and disadvantages.) What are uses to which networks are applied? What are the risks that someone might gain access to data from credit cards, home control software or medical records? What are the benefits of having easy ways to purchase, turn off the oven from work or share medical records quickly in an emergency? Responses will vary, but students should be encouraged to think beyond the obvious answers popular in the media. If privacy is a concern, what are the balancing benefits? How do we build checks and balances into automated systems? Does trying to stop technological development (such as cloning) actually stop it, or does it just move it to other countries where there may be no controls at all?

INTERMEDIATE
In the early days of computers, intelligence resided in a single enormous machine. Later desktop PC's replaced most large central servers. Why do you think the distributed intelligence of desktop PC's won out over centralized intelligence? What role does networking play in making distributed intelligence feasible? Response time from a central computer that serves hundreds of users is typically much slower than a PC dedicated to use by a single user. As the costs of computers dropped, the advantage of having control over the files, settings and accessories on your own PC also made desktop PC's much more desirable. With networks, many of the file-sharing and communication advantages of large computers became possible on desktop PC's as well.

ADVANCED
Set up a WorldLink server (or two for more fun) and connect to robots. Let everyone in your classroom, school or district know when the robots will be available to operate. Copy WorldPass from your ActivMedia Robotics Basic Suite CD and launch it. Make a presentation in front of your online robot, describing and demonstrating what you've learned about robotics, sensors and networks. Then encourage them to use WorldPass to try out your robots themselves to drive, fetch, Chat and take Snapshots. You'll need to either allow everyone anonymous guest access or set up an ID and password for them to use. (See Using ActivMedia Robots over the Internet, page 5 for details.) If you have an accurate map, load it and allow remote users to drive the robot around the map.

RESOURCES (The following discuss benefits and difficulties created by the Internet and networking.)


Chapter 6

COMMUNICATING ACROSS CULTURES

In today's global society, the ability to communicate -- in every sense of the word -- becomes a critical skill for success in both career and personal life. As adults, today's youth will need to be able to work with people of many cultures and languages. Robots like AmigoBot provide an easy way to practice communication skills with people of other languages and cultures no matter how far away.

REMOTE COMMUNICATION: FRIENDS AROUND THE WORLD!

Many classes have pen pals, but with AmigoBots, students can actually interact with other classrooms around the world! This opens a whole new realm of possibilities for team projects, foreign language, social science instruction, and remote learning.

Take advantage of the wonderful ability for cross-cultural activities provided by a mobile videoconferencing system in AmigoBot. The following project provides a structure for interacting so students from very different cultures can learn about each other first-hand. This structure will help overcome initial barriers and reduce embarrassment or fear of saying something "wrong." At the same time, it maximizes learning by letting students observe each others' demeanor and see each other's environs, adding context and personality to a subject that too often has to be taught in a vacuum. And, when students feel comfortable, they can move to a less structured Chat with partner students. The lesson that follows emphasizes foreign language study, but creative social science teachers can easily modify it to fit their purposes or use it to structure a thematic unit. This lesson is specific to AmigoBots, particularly those with ePresence systems.

LESSON 5: COMMUNICATING WITH AMIGOSOUNDS & WORLDLINK

GOALS
- To partner in teaching and learning second languages with native speakers of another tongue
- To learn to use AmigoSounds to customize a playlist of sounds on a robot

WHO?
Middle school - adult

HOW LONG?
THREE SESSIONS: Students will need a 30-40 minute session to experiment with existing sounds, understand the concept behind a playlist. They also need to begin thinking of words and phrases they would like to include in their own playlist. A second 30-40 session should finalize the words and phrases and record them. The third 30-40 minute session will be an opportunity to try out their list with a partner class, playing a game.
WHERE?
In a classroom or lab. Robots should be within 100 feet of the PC’s running them.

PREPARATION

1) Select a classroom with whom to partner. Penpals (see Resources, p. on page 32) or a sister classroom of students whose native tongue you’d like to study are a perfect choice. You might also partner with a school studying a similar or complementary topic or aspect of culture. If no partner school is available, students may practice with each other from different computers in the same classroom.

2) Download ActivMedia Sound File sample for this lesson, LanguageStudySample.ams, from http://robots.amigobot.com to a place on your network or PC’s where students can access it. The ActivMedia/WorldLink/Sounds directory would be a good place.

3) Before Session 2, locate a microphone. Nearly all notebook computers have built-in microphones, in case you don’t have access to one that will plug into your classroom PC. If you have a microphone, but the plug is too large to fit into your PC, ask your computer lab or AV department for an adapter (RC style to PC audio). Any Radio Shack or similar store should sell them.

4) Next find a sound tool to record sounds in .wav or other acceptable formats. (See the AmigoBot User’s Guide for more details on recording formats.) The Sound Recorder software in the WIN98 or WINME Start Menu under Programs/Accessories/Entertainment is okay. Set the sound properties for recording to PCM 11,025 MHz, 8-bit Mono. In WIN98/ME Sound Recorder, you may want to save this sound file format as the AmigoBot format and Convert Now so that all future files will be saved in that format.

5) Before Session 3, if your partner school does not have a robot, direct them to download ActivMedia WorldPass software from http://www.mobilerobots.com, so they can connect to your robot. You will also need to set them up as an accepted user in WorldLink and give them your computer’s IP address, optional ID and password so that they can connect to your robot.

ORGANIZING
Divide into 2-4 teams in a single classroom or have half the teams at a partner school. The ideal set-up is one PC, one AmigoBot ePresence system (including tether) and one microphone per team, but of course teams may rotate equipment.

ACTIVITIES

SESSION 1
1) Have your students launch WorldLink or Navigator and connect to a robot. Pull down on the Sounds menu and choose Show Sounds to see the Play List.

(FI SOUNDS ON IS INACTIVE, WAIT A MINUTE AND THEN TRY AGAIN; THE SOUNDS TAKE A BRIEF TIME TO DOWNLOAD TO THE MICROCONTROLLER AFTER CONNECTION.)
2) After students have tried out some sounds from the default Play List, have them launch AmigoSounds from the Tools menu in WorldLink or Navigator. Once in AmigoSounds, students can Load File. Choose default file AmigoBot.ams, from the default C:/ProgramFiles/Activmedia/Sounds directory.

3) Once they've opened the file, let students select various sounds in the AMS window and Play Sound so that they see that the sounds listed in the window are the same that they played on the robot earlier.

4) Next, explain that other playlists can be developed and used on the robot. Have students click Load File again and open LanguageStudySample.asm file that you downloaded (see Preparation).

5) What words and phrases are available for the robot to say, in this new playlist? Have students find objects around the classroom to match the phrases and set them in places that can be seen from the robot's camera or that the robot can drive to. Have them ask each other questions using the menu and answer by speaking into the robot's microphone.

6) Have students or teams ask the robot a question that can be answered from the playlist. For instance, "Donde esta' tu asiento?" (Where is your seat?) and then driving the robot to the chair and choosing, "Mi asiento" (My seat) and "esta' aqui." (is here.) from the playlist. This practice allows students to understand how they will be interacting with their partner classroom in another country. If they were designing such a list to teach other students English, what words and phrases would they include? Students may make their lists in either their own language, to teach the partner class, or in the language they're studying, for their own practice in conjunction with the partner school.

SESSION 2

1) Once students understand the concept, they can then design their own word and phrase playlists in Session 2. Before compiling their choices into a playlist with AmigoSounds, they will need to record their sounds with a microphone and any sound tool. See details in "Preparation," above.

2) Once students have created a sound for each word or phrase they want to incorporate in their playlist and have them stored in a directory, have them launch AmigoSounds. The default directory for ActivMedia Sound Files (.ams) is C:/ProgramFiles/ActivMedia/Sounds. Next students should Load File LanguageStudySample.ams again. Remove all User Sounds so students can replace them with their own.

3) To download new sound files onto the microcontroller, attach the long tether (ribbon cable) that came with AmigoBot, to the serial port under the robot. (Note that the serial connection on top of the robot is different and cannot be used to download sound files.) Then click on Help and choose Download from the Topics menu in AmigoSounds. Follow the few steps there to download the files. After about ten minutes, when the process is finished, students will be able to connect to the robot and test their files on the robot.

SESSION 3

1) Students are now ready to try out their conversational tools with a partner class that wants to learn their language. The Sounds menu provides an easy way for students to begin learning a little about each other and practicing their conversational skills without much risk of embarrassment or failure. By choosing items from the menu, they can begin to talk in a structured way with each other. It also allows interaction with partner schools that do not have a robot on their side. Students with robots can also speak via the robot's microphone.

Students in classrooms without robots will be limited to two modalities: phrases from the robot's playlist, and text typed in the chat window of the WorldPass software they've downloaded (see Preparation).

2) As students become more comfortable, they may begin to rely less on the existing list of phrases and move to typing questions into the Chat window. Encourage both sides in their efforts by speaking within range of the robot's microphone. Monitor conversation and intercede to assure that both sides
are doing equal amounts of speaking in each others' tongue. If students on one side are more fluent, the conversation could tend to become monolingual.

**CHALLENGE**

**ALL LEVELS**

Even though your partner school probably does not have its own AmigoBot, they may still want to create their own sound files in their language to run on your robot. After they send the files to your school, your students can create a playlist and download them onto a robot in your classroom. Then repeat the activities above, letting them control the robot's speech.

**RESOURCES**

The Teacher's Corner PenPal Requests, [http://www.theteacherscorner.net/penpals/index.htm](http://www.theteacherscorner.net/penpals/index.htm), to find penpals for your classroom

Chapter 7

COMMUNICATION AND TEAMWORK

As adults, today's youth will need to be able to work on distributed projects, collaborating with other teams in other places. Depending upon the distance and resources at hand, teams may communicate through limited modalities. Robots like AmigoBot, Pioneer and PeopleBot provide an easy way to investigate the advantages and disadvantages using various modalities to communicate information remote teams.

Figure 17: Vision, Speech and Audio Switches in WorldLink, WorldPass or Navigator

COMMUNICATION HAS MANY MEANINGS

To some people, communication refers to an interaction between people. In the world of technology, communication also means the ability to send information from a computer to a robot or from a camera to a person. Communication refers to the fact that information, images, sounds, thoughts or feelings are sent from one entity to another and that the other actually receives and understands them. Just receiving information does not necessarily mean that the person or machine understands anything that was transmitted, however. Computers don't understand the email messages they transmit and receive because they do not have the sensing and perception necessary to do so. If a person sends an email message to a person who speaks a different language, the reader can sense the text and maybe even perceive which letters of the alphabet were transmitted, yet they still may not understand the message.

Communicating remotely is different than communicating face-to-face in many ways. With telephones, tone of voice becomes very important. In email and online chat, emotions may be communicated with symbols like happy faces because words alone can be ambiguous. Remote communications may be delayed, so that what is seen actually happened several minutes before. In fact, the Mars Rover was so far away that communications back and forth to NASA headquarters were greatly delayed.

Over long distances, speed and bandwidth are important factors restricting communication. The distance to Mars is so great that communications traveling at the speed of light (186,000 miles per second) take twenty minutes even when Earth is at its closest position, 35 million miles away. While the speed of light is adequate for most communication on Earth, transmission can be slowed by message size. For instance, graphics or video, which are large data files, take longer to transmit than smaller files like text or even sounds. Because of these restrictions, people working together over long distances tend not to use the more bandwidth intensive media such as real-time video. If video is used, it may be a

VOCABULARY

Bandwidth - the amount of data that can be transmitted simultaneously through a communications channel

Communication - transmission of thoughts, feelings, information or sensation between entities

Distributed Project - complex projects comprised of tasks allocated to various individuals or groups that together must create the desired outcome; e.g., NASA Mars mission

Modality - type of sense, such as seeing, hearing, here used in the context of communicating through text, speech, sight or symbol
snapshot sent every few seconds. Dealing with lags and restricted channels of communications is part of the process of working together remotely.

With ActivMedia Robotics Basic Suite, students can experiment with remote communications using various modalities: only chat, only vision, only sound, only chosen phrases from a menu, or a combination of all these options. Which type of communication works for social conversation? For collaborative problem solving? For communicating a few facts? For emergencies?

LESSON 6: INVESTIGATING COMMUNICATION MODALITIES & REMOTE TEAMWORK

GOALS
- To discover which modalities facilitate which types of communication
- To experience some of the challenges of communicating and collaborating remotely

WHO?
Middle school - adult

HOW LONG?
30-40 minutes

WHERE?
In a classroom or lab; robots should be within 100 feet of the portable or desktop PC's running them.

PREPARATION
Draw a chalk path in the shape of a letter Z about four feet tall, as shown in Error! Reference source not found..

Launch WorldLink software on one PC per robot and test the connection by choosing Start WorldLink. This is Team 1’s PC.

Launch Navigator, WorldLink or WorldPass on a Team 2 PC for each robot. Choose Connect to WorldLink and follow the instructions for connecting to Team 1’s PC. See “Using ActivMedia Robots over the Internet”, page 5 for help in finding your IP address.

The Team 2 PC may be in the same room or at a different location as the robot and the first computer. Ideally, the teams should not be able to see or hear each other, except via the robot.

Make sure each robot has Settings: Safe Speed on. Click Take Control to test remote operation. Leave software running, but Disconnect from WorldLink and from the robots and turn them off.

ORGANIZING
Choose one person as scorekeeper. Divide remaining students into two teams per robot. Station one team (Team 1) at each robot's host PC. The second team for each robot (Team 2) should be at the remote PC. Ideally, the teams should not see or hear each other, except via the robot.
ACTIVITIES

1) Have Team 1 set the robot at one end of the chalk/tape path. Their job is to navigate from one end to the other. Instruct Team 1 to Connect to the Robot and drive from start to finish along the path using joystick or keys. How many seconds does it take?

2) Then have Team 1 Start WorldLink so that the robot is available to drive over the Internet. Have Team 2 launch Navigation, WorldLink or WorldPass software and Connect to WorldLink using the IP address of Team 1. Team 2 should then Take Control of the robot. First, time Team 2 driving along the chalk or tape path. Team 2 may watch the robot or its camera view and driving with joystick or keys. Is there any difficulty? In addition to the difficulty of teleoperation, there could be a delay between student input and robot response that makes it more difficult to drive remotely.

3) Now add another layer of complexity. Have Team 2 turn off video by clicking the camera icon in their display so they cannot watch the robot. Have Team 1 stand near the robot and issue verbal instructions to be transmitted to Team 2 via its microphone and WorldLink. How long does it take to drive the robot successfully from one end to another?

4) After one team has completed the course, rotate teams. Does practice improve times? What tactics do students develop in order to compensate for not being able to see the robot?

CHALLENGES

ALL LEVELS
Working remotely to accomplish a task can be difficult. It requires good communication skills and patience. Repeat Step 4 above, but turn off the microphone icon in WorldLink and have students communicate only via Chat typing. How long does it take to drive the robot successfully from one end of the course to the other?

Given teams' experience with this simple task, what kinds of problems might people experience when they're working together remotely on a large project such as a Hollywood film, a scientific study of global warming, or a new aircraft design? What are the strengths of each modality of communication? Full vision and sound, without delays, provide feedback that allows people to respond quickly and accurately in a situation like driving. When the feedback is delayed by a slow network connection, reacting appropriately is more difficult. When we lose visual feedback and only have sound, we can still hear from the emotion in peoples' voices how well we're doing. Their words tell us what to do, but words provide far less information about the robot's whereabouts than a picture. In Chat mode, we lose emotion as well, so we only have bare information, possibly augmented by smiley faces or frown symbols.
Chapter 8

Robots Extending Our Reach

At an early age, babies learn that they can extend their reach by grabbing a stick or other tool. Apes, birds and other animals use tools to dig or otherwise improve the capabilities of their bodies to act upon the environment.

Global Internet communications have empowered humans to reach far beyond the length of a stick. In fact, by turning senses into bytes and bytes into motion, people can see and act anywhere in the world where there is a computer and a sensing/manipulation device at the other end.

Figure 19: Students learning to use robots in a cooperative team

Remote Sensing: The World at Your Fingertips!

Students have already experienced real-time remote sensing. In this chapter, we let them investigate remote handling, teleoperating a robot. If robots have camera and surveillance or ePresence systems, have students teleoperate them in a hallway or a room that is outside their own field of vision. Using WorldLink or Navigator, practice operating a robot far from your location, via the Internet. What are the implications of humans being able to move objects from halfway around the world?

Remote sensing and manipulation are emerging technologies, still in their infancies. Top surgeons can already perform surgery from distant locations without having to delay operations by traveling to the operation site. Robots can handle dangerous medical or nuclear wastes without endangering humans. People will soon be able to use remote sensing to feed their pets, check the temperature of their homes, and turn on the oven to cook the evening meal - all without leaving work.

Lesson 7: Remote Handling and Cooperative Teams

Goal
To investigate how the Internet and other networking affect the people communicate and work together as well as how machines collaborate

Who?
Middle school - adult

How long?
Time: 30 - 40 minutes

Where?
Best conducted in an open space such as a gymnasium or a wide empty hallway outside a computer lab or classroom. Robots should be within 100 feet of the portable or desktop PC’s running them.

Vocabulary

Remote Sensing - seeing, hearing or otherwise monitoring aspects of an object or place too distant to be monitored unaided

Remote Handling - manipulating objects in a different location than the person handling them

Real Time - concurrently, with little or no delay
PREPARATION

This activity requires at least two robots. Test robots and connections if they have not been used recently.

Find one small brightly colored block and a length of standard 2x4-inch wood 4 feet long for each pair of robots. (2x4 will be below sonar level so the robots will approach the board, yet the board will be tall enough that the robots can push it rather than drive over it.)

In a place out of sight of the host PC’s (unless the robots have no cameras, in which case students will have to drive from the same room), on a smooth carpet or bare floor, draw a circle with chalk or tape and put a large dot in the center. The circle’s diameter should be three feet or less. Set the robots, wood and block outside the circle, as shown. Launch ActivMedia WorldLink or Navigator.

ORGANIZING

Divide the class into 2 or 4 teams. One robot and one PC per team works best, but teams can take turns. If taking turns, don’t let later groups see the first two teams working or they’ll know the solution.

ACTIVITIES

1) The activities in Lesson 5 were designed to help students understand some of the technology behind networking, remote control and remote sensing. Lesson 7 is designed to make clear some of the implications of our expanded powers due to networking. How do people work together remotely? How is communication affected over very long distances?

Figure 20: Set-up for a “hazardous waste” handling activity

2) Inform students that the block symbolizes a canister of leaking radioactive waste. If students touch it with their robot, it will eat away the robot’s body and incapacitate it. Likewise, their robot will be out of commission if its wheel touches or drives into the waste containment circle. Their job is to move the nuclear waste into the pit - the circle - for containment. To do so, they will have to drive the robots remotely. How can they accomplish their goal?

Figure 21: A cooperative solution to the hazardous waste handling problem

Teams will probably discover that cooperating is the easiest way to accomplish the task. A team may try to push hazardous waste into
the containment circle by poking it with the long end of the wood, but by far the easier way is shown in Figure 16. If students are highly competitive, they may need some help from the teacher in seeing that cooperation is the key to success.

CHALLENGES

ALL LEVELS

In the cooperative activity above, you learned that people need to collaborate with each other to accomplish some tasks; but did you also notice that both teams were collaborating with the robots. For instance, what happened when you tried to drive toward the other robot or toward a wall during the activity? The robots are programmed to override people's commands if the commands would result in their crashing. But how does a programmer decide when to let the robot ignore human commands?

Read the essay in Resources, below, which discusses how machines and people work together. Think about the scenario described at the end of the essay: how would you decide whether to let the robot or the human take control?

RESOURCES

"Will Robots Take over the World?", www.amigobot.com/amigo/future.html
Chapter 9

INTRO TO ROBOT COMMAND

Knowledge is power is an adage that rings truer today than ever before. Students who understand the scientific and technological processes underlying the Information Age are better prepared to control technology, rather than to be controlled by it. Whether students become engineers or artists, their abilities will only be enhanced from that foundation. Commanding robots to sense and react to the environment builds understanding of how machines, animals and humans all respond to their surroundings.

Figure 25: Trainer window is an easy way to learn Colbert commands

ROBOT CONTROLS AND THEIR RELATIONSHIP TO SENSING

The Trainer module in ActivMedia Robotics Basic Suite offers three ways for students to control robots. The simplest is to type commands directly into one or more of the five boxes on the upper right side of the Introduction to ActivMedia Trainer screen. The second way is to edit demo activities in the Colbert Program window on the left side of the screen. The third method is for students to write their own robot activities in the Colbert Program window. (A fourth method for advanced students, not covered in this guide, is to write and compile robotic behaviors in C/C++.)

LESSON 8: INTRODUCTION TO ROBOT COMMAND

GOAL
To let every student experience reactive programming, uniquely suited to mobile robots, in which a command statement links robotic behavior directly to its sensing of the environment.

WHO?
Middle school - adult

HOW LONG?
TIME: about 40-50 min

WHERE?
Best conducted with robots in an open area in the same room as the computers hosting them.

PREPARATION

ADVANCE PREPARATION
Launch Trainer from the main Basic Suite window. Connect to a Robot and type values into the Direct Colbert commands. Then Connect to a Simulator and type the same commands. This will give you a feel for the reaction of the robot. Note that distances are in millimeters, so numbers should be fairly large.

MATERIALS

- 1-4 Amigo Bots or other ActivMedia robots
- ActivMedia Navigator or WorldLink software
- 1 PC for each robot
BEFORE BEGINNING
Launch Trainer software on each PC and test the connection with each robot. Before class begins, leave software running, but Disconnect from the robots and turn them off.

ORGANIZING
Divide the class into one team per computer. One robot and one PC per team works best, but teams can take turns, and some can use simulators instead of real robots.

ACTIVITIES
1) Launch Trainer from the main Basic Suite menu. Turn on robots and Connect to them. Some students may Connect to Simulator instead of real robots.

2) There are five boxes in Trainer. Assign one or two boxes per team. Ask the teams to enter values, large and small, into the box, click Enter on the computer keyboard and watch what the robot or simulator does. Try large and small values to find which make the robot move well within the space allowed. When teams have finished the discovery process, have each explain to the class the function of their assigned box(es) and the size values that work best.

Figure 26: Colbert Direct Commands

- **Move (mm):** moves the robot forward the specified number of millimeters and then stops
- **Turn to (degree):** rotates the robot to a fixed heading of x degrees, from a zero heading facing directly right; the robot takes the shortest path, so if you choose a heading of 270, it will turn 90 degrees to the left. If you choose 360, it will sit still, since the heading doesn't change.
- **Turn (degrees):** rotates the robot by x degrees relative to its current heading. However, if the amount is greater than 180 degrees, the robot will take the shortest route to its goal, turning counter-clockwise instead of moving clockwise the specified amount.
- **Speed (mm/sec):** sets a constant forward velocity; the robot will run at that speed until it is stopped.
- **Rotate (deg/sec):** sets a constant rotational velocity; the robot will continue to rotate until it is stopped.

3) A demo activity automatically loads in the Trainer window on your right. Students running Trainer with the simulator should also click File: Load Map and choose ActivMediaLab.wld. This loads a "world" for the simulated robot to sense. Also, click File/Open and choose Demo 0. Demo 0 is a very short activity. Each line is followed by a comment, or explanation inside the symbols /*...*/. Have teams click Run and watch what happens. Can they explain what the robot is doing?

4) Explain that Colbert Activities are similar to C/C++ programs. All Colbert activities begin with act (name). A left curly brace { begins the first statement. While statements continue to execute until
their value is 0. Since the value of `while(1)` will never be zero, this statement executes forever. Inside the `while` loop, the activity reads Sonar 3 and moves forward the distance read by the sonar minus 400 mm (Sonar 3 is the fourth sonar from the left or the front-right sonar, since sonar are numbered beginning with 0). The right curly brace `}` ends the statement. `Start react();` runs the program.

**Figure 27: Demo 0 - React Activity in ActivMedia Trainer**

```c
/*
 Demo 0 is an example of a very short program
 that drives the robot toward a wall and then backs
 away from the wall. Consider the command:

 move(sfSonarRange(3)-400)

 Can you determine why the robot stops going
 forward and begins to back up even though it's
 only following this same command over and over?

 NOTE: When running Demo 0 with the simulator,
 choose Load Map/office.wld from the File menu so that
 the robot has sensor readings to control its behavior.
 Otherwise, it will just drive forward endlessly.

 */
 act react() /* this names activity 'react'*/
 {
  /* an open curly brace begins the activity*/
  while (1) /* while 1>0 (i.e., forever) */
  {
   /* curly brace starts the while loop */
   move(sfSonarRange(3)-400);
    /* read Sonar 3's distance from the nearest
     obstacle; then, move the robot that
     amount - 400 mm */
  }
  /* closing curly brace ends the while loop */
 }
 /* closing curly brace ends the activity */
 start react(); /*now start the activity*/
```

**CHALLENGES**

**BEGINNER**

Can you think of a machine or appliance around your house that translates a sensor reading into physical action? A heating system has thermostats that sense room temperature. Temperature sensing may done with a mercury switch. If the mercury heats and expands beyond a set level, the spring balance of the switch tips, completing a circuit that causes the furnace to fire up. When the mercury cools and contracts, the switch tips back in the opposite direction, extinguishing the furnace fire. Refrigerators, automatic lights, motion detectors and dryers are other devices that translate sensory input into action.
INTERMEDIATE
Experiment further with Direct Commands. What happens if teams enter a Speed between 100-300, followed by another command? What happens if Rotate is followed by another command? If Turn or Turn To is chosen after Speed, the robot will turn as it is moving forward, causing it to curve. If Move is chosen after Rotate, the robot will move forward as it rotates, causing it to circle while spinning.

ADVANCED
Demo 0 is interesting because it directly translates a sensor reading from the environment into an action by the robot. This should help students to understand how robots and other machines respond to the world around them. This type of programming is called reactive or behavior-based programming. Behavior-based programs respond to their environments in direct ways that can simulate insect intelligence. See Demo 6, BugBot, for a slightly more complex reactive program that relates a robot’s velocity to its distance from an obstacle. The bug in Demo 6 is not a terribly bright fellow; it takes it quite a few tries to get around an obstacle. What can students think of to do that would decrease the number of times the BugBot runs into the wall? If students are experienced programmers, their first reaction will be to add some logic to the program with if statements. Try to get them to understand that insects and reactive programs use equation values rather than logic trees to control the robot’s responses.

First they should analyze Demo 0. Why does the robot begin to move backward in Demo 0? Because the value of sfSonarRange(3) falls below 400 so the total expression becomes negative. Now look again at Demo 6. What happens to velocity as the robot approaches the wall? As it back away? Try adding other sonar readings or using ratios between sonar readings or other mathematical relationships to determine the movements.

RESOURCES
Appendix B of this Guide: Colbert Functions, Constants and Variables
Although interesting and fun for a while, Direct Colbert Commands only show how to move a robot in one direction at a time. The Colbert real-time programming language lets students do more. Colbert programs include multiple commands and logic to make robot behaviors, which in Colbert are called activities. In their activities, students can check the state of sonar or other elements of the robot, and make the robot respond to the environmental features it encounters as it moves.

**COLBERT FUNCTIONS**

A Colbert activity is a sequence of statements that the Colbert Evaluator reads, interprets, and executes in order. A Colbert activity begins with its name, followed by an opening curly brace {. The brace is followed by all the program commands, loops, variables, and so forth – followed by the ending curly brace }. Beyond the curly brace that closes the last activity, at the end of the text, you may add a series of commands that instruct the Colbert Evaluator how and when to execute the included activities.

Between the curly braces, you insert commands for the robot to follow. You already know some of these commands, since you used them in the Direct Colbert Commands section of Trainer. The direct commands Move, Turn To, Turn, Speed, and Rotate are all examples of functions in Colbert. When programming an activity, you simply add a set of parentheses () after the name, inserting a value within the parentheses. For example, “Move” with an entered value of 1000 from the Trainer’s Direct Commands section translates into move(1000) within a Colbert activity file. A full description of Colbert functions is located in “Appendix B: Colbert Commands, Operations and Variables”, page 62.

**COLBERT DEMOS**

Figure 30: Demo 1 - Square shows how to create a Colbert activity. The activity, named “square,” has code that, not surprisingly, drives the robot in a square. Before you can run the program, you will need to Connect to either a real Robot or the robot Simulator.

As you can see, the first line of Demo 1 starts with the keyword act, which describes it as an activity; Colbert activities must start with this keyword. Following the keyword is a name for the activity. Names should describe what the activity accomplishes. In this case, our activity drives the robot in a square, so we named the activity square. The parentheses after the name are for values of entering arguments. Though we do not use entering arguments in this guide, we include parentheses because it is good programming form.

Notice in the first line the text Demo 1 moves the robot or
simulator in a square. is enclosed within the /* and */ characters. These are known as comments in programming. A comment is a description of what a certain piece of code does and is very useful if you have a large program or activity. The Colbert Evaluator will ignore any code separated by these characters, as they are only needed for the programmer to remember what the code does.

Figure 30: Demo 1 - Square

```cpp
/*
Demo 1 moves the robot or simulator in a square.
Connect to the robot or simulator above, then
click "Run" below and watch the robot go.
*/

/* Phrases that begin and end like this are comments
and do not affect the way the program runs. */

act square() /* This line names the activity "square" */
{
    /* The curly brace opens the activity */
    int a; /* This line sets up integer variable "a"
    Note that program lines other than
    loops need to end with semicolons */
    a = 4; /* This line sets the value of "a" to 4 */
    while(a) /* This line begins a while loop that continues while a > 0 */
    {
        /* This curly brace opens the while loop */
        a = a - 1; /* This line reduces the value of "a" by 1 */
        move(1000); /* This line moves the robot forward 1000mm */
        turn(90); /* This line turns the robot 90 degrees */
    } /* This curly brace ends the while loop */
    /* This curly brace ends the activity */
}

start square(); /* This command runs the activity "square" */
```

The next line of the activity is a curly brace { that indicates the beginning of this activity. All activities have a body, the area separated by the left and right curly braces. Anything within these braces is part of the activity, and will be read by the Colbert Evaluator. Curly braces are also used to separate the contents of a loop, such as the while loop in Demo 1.

The third line of code in Demo 1 sets up a variable. A variable is an item that may change value during a program. This activity has one variable, a, which will have subtraction performed on it later in the activity. The first thing to do to a variable is set its numerical type. For counting the number of loop repetitions, we will use an integer variable. In Colbert programming, an integer is any number, usually in the range of -32,767 < 0 > 32,768. The third line of code sets the variable a as an integer, using the keyword int.

After assigning the variable type, you should set its value. In this activity, we set the initial value of a = 4 since a square has four sides. Now, the variable a has the exact value of 4, so we can calculate with the variable just as if it were the number 4.
The next section of the activity is a **loop**, specifically a **while loop**. Loops repeat a certain task until the value of the loop’s argument is false. In computer programming, false = 0 or NULL. In Demo 1, the **while** loop runs until the variable \(a = 0\). Because we set \(a = 4\), the robot will repeat the loop four times, which happens to be the number of sides in a square. If we want the robot to drive around the square twice, we would set \(a = 8\).

Every time the activity repeats, the fifth line of code \(a = a - 1\) decreases the value of \(a\) by one so that it will eventually equal zero. When \(a\) equals zero, the **while** condition will become false and the program will move on instead of repeating the loop again.

Since there is no code following the **while loop** (between the **while** loop’s ending curly brace and the activity’s ending curly brace), the robot will stop moving. The closing curly brace, `}` tells the Colbert Evaluator that the activity has no more instructions, and the Colbert Evaluator stops running.

The final line of the activity file, `start square`, is not part of the activity itself. It is a line translated by the Colbert Evaluator to start program execution as soon as the file is loaded. Thus, you may store more than one activity in a file, but only start one of them. The second activity might be called if the robot reaches a certain state.

---

* Colbert states are beyond the scope of this guide. For more information on states, see Chapter 7 of the ActivMedia Software Guide.
As shown in Figure 32, Demo 2 starts out the same as Demo 1. But at the end of each move, the robot turns 180 degrees instead of 90. This makes the robot move back and forth like a sentry instead of in a square.

**Lesson 9: Beginning to Program in the Trainer Window**

**Goal**
To familiarize students with Colbert functions and command structure.

**Who?**
High school or college-age programming students; advanced middle school students.

**How Long?**
**Time**: about 40-50 minutes

**Where?**
Best conducted with robots in an open area in the same room as the computers hosting them.

**Preparation**

**Advance Preparation**
Launch Trainer and familiarize yourself with Demos 1 and 2.

**Before Beginning**
Launch Trainer software on each PC and test the connection with each robot. Before class begins, leave software running, but **disconnect** from the robots and turn them off.

**Organizing**
Divide the class into one team per computer. One robot and one PC per team works best, but teams can take turns and some can use simulators instead of real robots.

**Activities**

1) Have students turn on the robots and connect to them. Then review Direct Colbert Commands such as **move** and **turn**.

2) Have students **load** Demo 1 and click **Run**. What two commands can they change to make the robot move in a pentagon? They would need to edit line 13 to read: a=5 and line 18 to read **turn(72)**. Why should the robot turn 72 degrees? Because 360 degrees / 5 sides = 72 degrees per angle. What would be the **while** argument and **turn** argument for an octagon? a=8 and **turn(45)**.

3) Have students load and try out Demo 2. After they have observed the robot or simulator’s actions, ask students to explain the program or explain yourself step by step how it functions. Be sure to mention the importance of signals like comment symbols, semi-colons, curly braces and why the Colbert Evaluator needs to see them in order to interpret the code properly.
4) Have students **Load and Run Demo 2.** How might they alter Demo 2 to make the robot drive in more interesting patterns?

**CHALLENGES**

**BEGINNER**
Can you create a program that will drive a robot in the shape of a figure 4, as shown at left?Hint: How many degrees must the angles of a triangle always equal? For help, see www.amigobot.com/amigo/demos.html

**INTERMEDIATE**
Can you create a program that will drive a robot in a figure eight? For help, see www.amigobot.com/amigo/demos.html

**ADVANCED**
Can you create a short program that will drive a robot in a circle continuously? What happens when you change the parameters of the speed function or the rotate function? For help, see www.amigobot.com/amigo/demos.html

**RESOURCES**
Chapter 11

Responsive Programming

The difference between intelligent robots and most machines is that intelligent robots can sense aspects of their environment and respond in different ways, depending on the conditions. For example, intelligent robots typically try to avoid obstacles they sense in their path. Using vision, lasers, GPS or other sensing devices, intelligent robots measure various aspects of the conditions around them. Responsive computer programs base their actions on those conditions.

Two Approaches to Responsive Behaviors

Two basic approaches to responsive programming are reactive and deliberative. One type, mimicking the insect world, tries to “hardwire” or directly relate inputs to outputs. This results in short programs like Demo 0 from Chapter 9. In Demo 0 the sonar range output is used as the basis for the speed parameter input. Because these programs naturally adapt to the environment without requiring any analysis of the input by the robot, they are called reactive. Other responsive programs, such as Demo 3 below, read sensor data, analyze it and then logically determine a course of action. This approach is called deliberative.

Many roboticists combine both approaches when designing and building responsive systems. Typically, they use behavior-based, reactive approaches for low-level behaviors like obstacle avoidance, and deliberative approaches for creating higher-level applications.

Programming Robots to Respond to Their Senses

Demo 3 is an example of a deliberative activity that polls all the robot’s sonar as it drives a robot forward. It analyzes each reading to determine whether it is closer than the allowable limit of 400mm. If any sonar reading is less than 400mm, the robot stops.

Figure 35: Demo3 moves at speed 200 until it senses an obstacle

```c
/*
Demo 3 instructs the robot to move until it senses an obstacle, at which time it will stop
the robot. This is a very simple example showing how to use the sonar readings for obstacle
avoidance. Can you modify the program so that the robot drives moves away from an
obstacle instead of simply stopping? (See Demo 5 for hints.)
*/
act AvoidObstacle()
{
  int clearpath; /* set up integer variable named clearpath*/
  speed(200); /* move the robot at a constant speed of 200mm per second*/
  clearpath=1; /* set indicator to true*/
```
The first part of this activity declares a new variable `clearpath`, which is used in the while loop. The next step of the activity is to start the robot moving, in this case `speed = 200 mm/sec`. Then the `clearpath` variable is set to 1 (true), since the robot does not yet sense an obstacle. Next we enter the while loop, which continuously checks to see whether `clearpath` is true (>0). Within the while loop, we use the `(sfSonarRange(sonar number) < 400)` function to determine whether the robot comes within 400 mm of any obstacle.

If an object is detected, we set the speed of the robot to zero with `speed(0)`, and set the `clearpath` variable to 0 (false). When false, the robot stops, and the activity ends.

Demo 3 is the beginning step for an obstacle avoidance activity. In fact, it contains intentional flaws so that students can refine this example in a number of ways to make it perform more robustly, as suggested in the Challenges that follow this lesson.
LESSON 10: RESPONSIVE PROGRAMMING

GOAL
To teach students how to make the robot detect obstacles and avoid them.

WHO?
High school or college-age programming students; advanced middle school students.

HOW LONG?
TIME: about 45 min

WHERE?
Best conducted with robots in an open area in the same room as the computers hosting them.

PREPARATION

BEFORE BEGINNING
Launch Trainer software on each PC and test the connection with each robot. Before class begins, leave software running, but Disconnect from the robots and turn them off.

ORGANIZING
Divide the class into one team per computer. One robot and one PC per team works best, but teams can take turns and some can use simulators instead of real robots.

ACTIVITIES

1) Have students launch Trainer and load Demo 3. Then they should connect to a real Robot or software robot Simulator. If they use a simulator, they should also Load Map under the File Menu. Why do they need to load a map to use Demo 3? If they do not load a map, the simulator will have no obstacles to provide sonar information, so all sonar readings will remain at maximum levels of 3000mm.

2) Have students run Demo 3. What is the robot doing? The robot stops when any of its sonar read less than 400. This deliberative program uses logic to determine whether it should tell the robot to stop. What symbol indicates a logical operation? The || symbol. What does || mean? It means OR. What is the difference between a logical OR statement and a logical AND statement? A logical OR statement means that the condition is true if any of the expressions is true. An AND statement means that all the expressions must be true for the condition to be true.

3) How is this different than the way the robot detected an obstacle and responded to it in Demo 0? Which demo was reactive? Which was deliberative? Demo 0 used the value from its sonar as the argument for its move command. Without any logic, this made the robot automatically back up when it approached a wall too closely. Demo 3 compares the sonar value to a minimum, 400mm, and stops the robot if the value is too low. Demo 0 was reactive; Demo 3 was deliberative.

4) Have students improve Demo 3 according to their abilities, as suggested in the Challenges below.

MATERIALS

• 1-4 AmigoBots or other ActivMedia robots
• Trainer software
• 1 PC for each robot
CHALLENGES

BEGINNING
Modify Demo 3 so that after it polls a sonar, it turns away from the first one that shows a level under 300mm. Can you be first in your class or team to create an activity that drives away from obstacles instead of stopping? If you have trouble, see Demo 5 for ideas.

INTERMEDIATE
Two lines of Demo 3 code are totally unnecessary for AmigoBot. In fact, they could cause the robot to stop when it was in no danger at all of hitting an obstacle. Which two lines are these?

\[
\begin{align*}
& (\text{sfSonarRange}(6) < 400) \quad \text{||} \\
& (\text{sfSonarRange}(7) < 400)
\end{align*}
\]

The above two lines read AmigoBot's rear sonar. Since the robot only moves forward in this program, information from the rear sonar is irrelevant and could actually cause the robot to behave inappropriately. Pioneer robots, which have 8 front and side sonar, need these lines.

Can you remove these two lines and still make the program run? Students will need to remove both OR symbols but leave the final ) ending the logical expression in order for the program to parse properly.

Also, sonar 0 and Sonar 5 are on the sides of AmigoBot; sonar 0 and 7, on Pioneer. It's beneficial to check side sonar, but setting the range to 400mm on the side limits the robot's ability to enter narrow spaces. The minimum range the sonar can sense is 150mm (6 inches). Try reducing the range of side sonar to 200mm so that the robot can function more flexibly. This is for AmigoBot:

\[
\begin{align*}
& \text{if} \ (\text{sfSonarRange}(0) < 200) \quad \text{||} \\
& \quad (\text{sfSonarRange}(1) < 400) \\
& \quad (\text{sfSonarRange}(2) < 400) \\
& \quad (\text{sfSonarRange}(3) < 400) \\
& \quad (\text{sfSonarRange}(4) < 400) \\
& \quad (\text{sfSonarRange}(5) < 200))
\end{align*}
\]

ADVANCED
One of the keys to becoming a good programmer is knowing how to debug. What statements could you add to Demo 3 if you were trying to debug it?

\[
\begin{align*}
& \text{sfsMessage} \ (d\%, \text{SonarRange}(0)) \quad \text{etc.} \\
& \quad \ldots \text{sfsMessage} \ (d\%, \text{SonarRange}(7))
\end{align*}
\]

Adding the above code before the final curly brace of the if statement would allow you to see the sonar readings displayed as the program was running.

Once you've added read-outs into the program, try expanding the program to make the robot follow the edge of the obstacle it finds. Can you approach the obstacle to a certain distance, turn 90 degrees and then try to follow the wall or other surface by maintaining the same side sonar reading as you travel?

RESOURCES

Appendix B: Colbert Functions, Constants and Variables
Robots need to know when their batteries need recharging, when their motors are in danger of burning out, and so on. Monitoring the sensors that help the robot determine its own state is called proprioception.

**Proprioception: How Robots Take Care of Themselves**

Demo 4 shows how to gather proprioceptive data from some of the user-accessible sensors of ActivMedia robots. These include encoders, motors, battery and sonar. Proprioceptive data allows the robot to poll the status of its own components to determine whether it needs maintenance or is involved in a dangerous situation. Students have already monitored sonar data to prevent the robot and objects in its environment from harm. They will learn how to check for a stalled motor, which if caught in a child's long hair or trapped in the fringe of a woven rug, could cause the robot to hurt someone or burn out its motors. They'll also see how to determine when the robot's battery needs to be recharged.

```
/*
   Demo 4, shows how to access many variables that reflect the current state of a robot. Then it moves the robot forward so students can see how the variables change.

   Robot status variables are listed completely in Appendix B of the Educators' Guide to Robots.

   NOTE: When using this program connected to the simulator, be sure to Load Map/office.wld under the File menu. Without a map, there will be no obstacles for the simulated robot to sense.
*/

act DoPackets()
{
  int battery;
  speed(200);
}
```

* Status variables are also used in determining robot states, very important to robot programming. For more information on robot states, see the ActivMedia Software Guide.

**Vocabulary**

- **Heading** - direction the robot is facing, polled by the variable named theta
- **Poll** - to check on the status of some component or sensor
- **Proprioception** - sensing one's own state or condition
- **Status** - current state or condition
- **Translational** - motion along a path
/* The following statements show how to display the status of some of the proprioceptive info available from the robot. For a complete list of status variables and what they indicate, see the Appendix of the Educators' Guide to Robots.

The Robot variables are already set up, so they don't need to be initialized. However, when their values are displayed with an sfSMessage command, you must specify the type of variable. (%d = decimal integer; %f = floating point) If the variable is floating point, you must specify the number of decimal places: %.1f = a floating point variable with 1 decimal place.

*/
sfSMessage("bat:%d", sfRobot.battery);
sfSMessage("ax:%.1f ay:%.1f ath:%.1f dist:%0.2f",
    sfRobot.ax, sfRobot.ay,
    sfRobot.ath, sfRobot.dist);
sfSMessage("tv:%0.1f rf:%0.1f leftv:%0.1f rightv:%0.1f",
    sfRobot.tv, sfRobot.rv, sfRobot.leftv,  
    sfRobot.rightv);
sfSMessage("acc:%0.1f mtv:%0.1f, mrv%0.1f",
    sfRobot.acc, sfRobot.mtv,  
    sfRobot.mrv);

/* what are the readings from each of the sonar */
sfSMessage("Sonar Range: 1:%d  2:%d  3:%d  4:f:%d",
    sfSonarRange(0), sfSonarRange(1),  
    sfSonarRange(2), sfSonarRange(3));
sfSMessage("Sonar Range: 5:%d  6:%d  7:%d  8:%d",
    sfSonarRange(4), sfSonarRange(5),  
    sfSonarRange(6), sfSonarRange(7));

/* move the robot, so you see some of the variables change on the next print */
move(1200);
sfMessage("Reprinting values now that we've moved the robot.");
sfSMessage("bat:%d", sfRobot.battery);
sfSMessage("ax:%.1f ay:%.1f ath:%.1f dist:%0.2f",
    sfRobot.ax, sfRobot.ay,  
    sfRobot.ath, sfRobot.dist);
sfSMessage("tv:%0.1f rf:%0.1f leftv:%0.1f rightv:%0.1f",
    sfRobot.tv, sfRobot.rv, sfRobot.leftv,  
    sfRobot.rightv);
Demo 5 shows how to use the status variables from Demo 4 to control the robot's behavior. The program moves about, checking for obstacles within 300mm of Sonar 2 and Sonar 3. If it finds an obstacle, it turns 90 degrees and then moves on if nothing is in front of it. Demo 5 also checks battery voltage. When it falls below acceptable levels (11.6v), the robot stops.

Be sure to load a map (e.g., File/Load Map/office.wld) if you are using this demo with a simulator, so the simulated robot will have walls and obstacles to sense.

Figure 38: Demo 5 - BatteryStop sets the robot wandering until it senses the need to charge its battery

```c
act BatteryStop()
{
    int robotstopped;

    /*
     * robotstopped tells me if I have stopped the robot
     * due to low battery
     */
    /*
     * now, initialize the variable, so it has a known
     */
```
value 0 == false  1 == true
*/
robotstopped = 0;
/

now, drive the robot around, until the battery
level is below 11.6 volts. If the voltage is that low,
I will stop the robot. You could program it to drive
to its Home location or to its docking station.
NOTE: The following simple obstacle avoidance
routine checks only AmigoBot front sonar, 2 & 3.
For Pioneers, change this to 3 & 4. If either
sonar senses an obstacle <300mm,
the robot turns 90 degrees. Can you
improve on this obstacle avoidance
*/
while (robotstopped != 1)
{
/*begin while*/
  sfSMessage("sonar3: %d sonar 2:%d",
    sfSonarRange(3), sfSonarRange(2)); /*These are AmigoBot front sonar*/
  if (sfSonarRange(3)>300 && sfSonarRange(2)>300)
    /*begin if statement 1*/
    move (100);
    /*move forward if no obstacle is sensed*/
  }/*end if statement 1*/
  if (sfSonarRange(3)<=300 || sfSonarRange(2)<=300)
    /*begin if statement 2*/
    turn (90);
    /*turn 90 degrees if an obstacle is sensed*/
  }/*end if statement 2*/
  sfSMessage("battery level:%d", sfRobot.battery);
  if (sfRobot.battery/10 < 11.6)
    /*begin if statement 3*/
    /*stop the robot*/
    speed(0.0);
    /*set robotstopped to true, so the while loop exits*/
    robotstopped = 1;
} /*end of if statement 3*/
}/* end of while loop */
}/* end of BatteryStop() */
start BatteryStop();
LESSON 11: PROGRAMMING ROBOTS FOR SAFETY

GOAL
To enable students to write activities that:

• gather proprioceptive feedback such as battery voltage level,
• use proprioceptive information to monitor the robot and design safety behaviors.

WHO?
Students with programming experience, upper middle school - adult

HOW LONG?
TIME: about 45 minutes

WHERE?
Best conducted with robots in an open area in the same room as the computers hosting them.

PREPARATION

ADVANCE PREPARATION
Familiarize yourself with Demo 4 and Demo 5.

BEFORE BEGINNING
Launch Trainer software on each PC and test the connection with each robot. Before class begins, leave software running, but Disconnect from the robots and turn them off.

ORGANIZING
Divide the class into one team per computer. One robot and one PC per team works best, but teams can take turns and some can use simulators instead of real robots.

ACTIVITIES

1) If any students know Spanish, ask them what proprio means. Self. Does anyone know or can they guess the meaning of proprioception? Sensing the condition of oneself. Today we're going to look at how robots sense and take care of themselves.

2) Have students load Demo 4 into ActivMedia Trainer and connect to a robot or simulator. Those connecting to a simulator should also Load Map/office.wld in the Trainer File menu. Each group should run Demo 4 several times with the robot or simulated robot in different positions.

3) Refer students to page 66, "Robot Status Variables." What is the meaning of each of the readings gathered in Demo 4? Robot.battery checks the battery level in volts. It's an integer even though it reads battery levels in tenths. Robot.ax, ay and .ath are the robot's x, y location and its theta heading. Robot.dist is the distance traveled. Robot.tv is the translational, or forward velocity; Robot.rv is the rotational velocity, or the speed the wheel turns. Robot.leftv and .rightv are the individual wheel velocities. Translational and rotational velocities may also have maximum allowed values: Robot.mtv and .mrv. Robot.acc is acceleration.

4) Why do the Robot. variables not need to be set up or initialized in Demo 4? Because they are built into Colbert and always available. What type of variables are each of the Robot. variables? As shown in Appendix B, except for battery level, all of these variables are floating point. What
statement polls a decimal integer like battery level?  \texttt{sfSMessage("\%d", sfRobot.battery);} polls the battery level. It prints:

\begin{verbatim}
11.8
\end{verbatim}

Anything in front of \$ is printed out literally. Thus, \texttt{sfSMessage("Battery level: \%d v", sfRobot.battery);} \texttt{would print:}

\begin{verbatim}
Battery level: 11.8 v
\end{verbatim}

What statement polls a floating point variable like \texttt{Robot.acc}?
\texttt{sfSMessage("\%0.1f", sfRobot.acc);} polls acceleration and prints it with one decimal place.

5) The next few lines read the sonar, as done in earlier demos. Do the sonar readings respond as students would expect when they move the robot into a corner? Into a hallway? Although sonar hitting at an angle to the wall may not be detected and sonar closer than 15cm (6in) to a wall may read at infinity, the readings should in general conform to the shape of the space around the robot.

6) Next have students run Demo 5. What is the robot doing? The robot is wandering, avoiding obstacles by turning 90 degrees. It is also monitoring its battery level and will stop when this falls below 11.6v. What does the logical symbol \&\& mean in the statement:

\begin{verbatim}
if (sfSonarRange(3)>300 \&\& sfSonarRange(2)>300)
\end{verbatim}

\&\& means AND, so when the Colbert Evaluator checks the truth of the if statement it tests for BOTH conditions to be true. Thus the range of both sonar 2 and 3 must be less than 300mm for the program to perform the body of the if statement (move 100). If you replaced the \&\& with the OR symbol, only one of the conditions would need to be true.

7) The robot is currently polling only sonar(2) and (3). Why were these two sonar chosen? Because they are the front forward two and the program only drives forward. Which sonar should you poll when driving backwards? Sonar (6) and (7).

8) Have students put a robot in an empty space at least two meters square. Then have them load and run Demo 5. Leave the robot running as students work on Challenges or other activities so they can watch what happens when the battery runs down.

**CHALLENGES**

**BEGINNING**

One of the keys to becoming a good programmer is knowing how to debug or fix a program that is not functioning properly. Which statements in Demo 5 help you debug the program or troubleshoot the robot in case of problems?

\texttt{sfSMessage("sonar3: \%d sonar 2:\%d", sfSonarRange(3), sfSonarRange(2));}

The first debugging statement lets you see the sonar ranges as they're read.

\texttt{sfSMessage("battery level:\%d", sfRobot.battery);} The second \texttt{sfSMessage} statement displays battery levels. How does printing sonar values and battery levels help with debugging? It lets you see the sonar values when a robot misbehaves and let's you confirm that the battery level actually is less than 11.6 if the robot stops.

**INTERMEDIATE**

Change Demo 5 so that it makes the robot drive at a velocity of 250 mm per second until it senses an obstacle within 400mm. Which statements would you need to change? New statements are shown below in green. Why is \texttt{WHILE} a more powerful and elegant way to program this behavior than using multiple IF statements? Using \texttt{WHILE} statements keeps the robot constantly monitoring its status rather than
requiring you to loop back to `IF` statements to continue checking. For a machine that is constantly running, like a robot, `WHILE` statements are usually more appropriate.

```cpp
act BatteryStop()
{
    while (sfRobot.battery/10 > 11.5)
    {
        while (sfSonarRange(3)>300 && sfSonarRange(2)>300)
        {
            speed (250);
        }
        sfSMessage("sonar3: %d sonar 2:%d",
        sfSonarRange(3), sfSonarRange(2));
        turn (90);
        sfSMessage("battery level:%d", sfRobot.battery);
    }
    speed(0.0);
}
start BatteryStop();
```

**ADVANCED**

When robots run into obstacles, either their bumpers sense the impact or their motors stall and fail to turn the wheels. Bumper readings (from Pioneers or PeopleBots with bumpers) and motor stall (all robots) are both read by ActivMedia robots for proprioceptive information. Write a program that will stop the robot if a bumper impact is detected or one of the motors stalls. (See "Appendix B: Colbert Commands, Operations and Variables", page 62, for the command to read bumper feedback and motor stall information.)

What other types of proprioceptive information might a robot gather? **Compasses, tilt sensors, water sensors and thermostats** are all examples of proprioceptive sensors. They provide proprioceptive information regarding compass heading, tilt and yaw, wetness and temperature.

The following are examples of gathering bumper or motor stall information. These activities do not stop the robot if the robot collides or is stuck; can you add the commands to make it do so? If your robot has no bumpers, use the motor stall value, but remember to test the stall value of both motors 1 and 2. Try using more elegant `WHILE` statements instead of `IF` to control the logic (See Intermediate Challenge above for more on `IF` vs. `WHILE`):

```cpp
act bumper()
{
    sfSMessage("Has bumper collided? %d", sfRobot.bumpers);
    /* Use ready-made variable sfRobot.bumpers to see if any bumper has collided. */
}
start bumper();
```

```cpp
act motorstall()
{
    int a; /* Since no ready-made variable for motor stall exists, initialize integer a*/
```
a = sfStalledMotor(1); /*Set a = motor 1's stall value*/
sfSMessage("Is motor stalled? %d", a); /*Print a; if it = 1, the motor has stalled.*/
}

start motorstall();

**RESOURCES** (for more advanced programming in Colbert)

Appendix B of this guide: Colbert Functions, Constants and Variables


<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 AD</td>
<td>Petronius Arbiter makes a doll that can move like a human being.</td>
</tr>
<tr>
<td>1557</td>
<td>Giovanni Torriani creates a wooden robot that can fetch the Emperor's daily bread from the store.</td>
</tr>
<tr>
<td>1921</td>
<td>The first reference to the word robot appears in a play opening in London. Czech playwright Karel Capek introduces the word robot from the Czech robota, which means a serf or one in subservient labor.</td>
</tr>
<tr>
<td>1942</td>
<td>Russian-born American science-fiction writer Isaac Asimov first uses the word robot in his short story, &quot;Runabout.&quot;</td>
</tr>
<tr>
<td>1951</td>
<td>Raymond Goertz designs a teleoperator-equipped articulated arm for the U.S. Atomic Energy Commission.</td>
</tr>
<tr>
<td>1954</td>
<td>George Devol designs the first programmable robot, and coins the term Universal Automation. He shortens it to Unimation, which becomes the name of the first robot company.</td>
</tr>
<tr>
<td>1961</td>
<td>General Motors installs the first industrial robot on a production line. The robot selected is a Unimate.</td>
</tr>
<tr>
<td>1962</td>
<td>The first industrial robot sees service in a General Motors car factory in Trenton, New Jersey. The robot lifts hot pieces of metal from a die-casting machine and stacks them.</td>
</tr>
<tr>
<td>1964</td>
<td>Artificial intelligence research laboratories open at M.I.T., Stanford Research Institute (SRI), Stanford University, and the University of Edinburgh.</td>
</tr>
<tr>
<td>1968</td>
<td>SRI builds and tests Shakey, a mobile robot with vision capability.</td>
</tr>
<tr>
<td>1970</td>
<td>Stanford University develops an electrically powered robot arm that becomes the standard for research projects. The arm is known as the Stanford Arm.</td>
</tr>
<tr>
<td>1973</td>
<td>Richard Hohn develops the first commercially available, minicomputer-controlled industrial robot for Cincinnati Milacron Corporation. The robot is called T3, The Tomorrow Tool.</td>
</tr>
<tr>
<td>1974</td>
<td>Professor Scheinman, the developer of the Stanford Arm, forms Vicam Inc. to market a version of the arm for industrial applications. The new arm is controlled by a minicomputer.</td>
</tr>
<tr>
<td>1976</td>
<td>Robot arms are used on the Viking 1 and 2 space probes. Vicam Inc. incorporates a microcomputer into the Vicam design.</td>
</tr>
<tr>
<td>1977</td>
<td>ASEA, a European robot company, offers two sizes of electrically powered industrial robots. Both robots use a microcomputer controller for programming and operation. In the same year, Unimation purchases Vicam Inc.</td>
</tr>
<tr>
<td>1978</td>
<td>Unimation develops the PUMA (Programmable Universal Machine for Assembly) robot, using Vicam techniques and with support from General Motors.</td>
</tr>
<tr>
<td>1984</td>
<td>First Personal Robot convention held in Albuquerque NM, featuring HERO robots from HeathKit and a personal robot from inventor of the video game, Nolan Bushnell.</td>
</tr>
<tr>
<td>Year</td>
<td>Event</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>1995</td>
<td>The Pioneer 1 Mobile Robot is introduced at IJCAI, dropping mobile robot prices by 80% so that many universities can afford intelligent mobile robots</td>
</tr>
<tr>
<td>1998</td>
<td>ActivMedia Robotics introduces Pioneer 2, adding built-in computing power previously available on only very expensive mobile robots</td>
</tr>
<tr>
<td>1999</td>
<td>Friendly Machines introduces their robotic lawn mower in the US, the most popular home appliance robot to date.</td>
</tr>
<tr>
<td>2000</td>
<td>ActivMedia Robotics introduces the AmigoBot and ActivMedia Basic Suite, making sophisticated intelligent robots easy enough for elementary school children to operate and program</td>
</tr>
</tbody>
</table>
APPENDIX B: COLBERT COMMANDS, OPERATIONS AND VARIABLES

VARIABLE TYPES*

The following table shows how to define program variables at the beginning of an activity before they are used. Variables used in only the current activity are called local variables; those used in more than one activity are called global variables.

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Example/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char (character)</td>
<td>char c; Defines the variable “c” to be a single byte of value 0 to 255 (-128 to 127).</td>
</tr>
<tr>
<td>int (integer)</td>
<td>int i; Declares “i” to be an integer variable, typically two bytes in size which may be any value from 0 to 65,535 (-32,768 to +32,767).</td>
</tr>
<tr>
<td>float [0.n]</td>
<td>float 0.1f; Defines “f” to be a floating point decimal variable, optionally with n decimal places; default is six. Note use of scientific notation: f = 1e-4 = 0.000100 or f = 1e5 = 100000.0000</td>
</tr>
</tbody>
</table>

* † Depends on computer and operating system, but typical of current PCs.

MATHEMATICAL OPERATIONS

<table>
<thead>
<tr>
<th>Operation</th>
<th>Meaning</th>
<th>Example/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Assign</td>
<td>a = 7; Assigns the value seven to the variable a. (Declared variable type should match the value; see Variable Types).</td>
</tr>
<tr>
<td>+</td>
<td>Add</td>
<td>c = a + b; Sets c equal to the sum of a plus b</td>
</tr>
<tr>
<td>-</td>
<td>Subtract</td>
<td>c = b - a; Sets c equal to the difference b minus a</td>
</tr>
<tr>
<td>*</td>
<td>Multiply</td>
<td>c = a * b; Sets c equal to the product of a times b</td>
</tr>
<tr>
<td>/</td>
<td>Divide</td>
<td>c = a / b; Sets c equal to the result of a divided b times</td>
</tr>
<tr>
<td>%</td>
<td>Modulus (remainder)</td>
<td>c = a % b; Sets c equal to the remainder after a is successively divided by b; for example, c = 3 for a = 21 and b = 6.</td>
</tr>
<tr>
<td></td>
<td>Bitwise OR</td>
<td>c = a</td>
</tr>
<tr>
<td>&amp;</td>
<td>Bitwise AND</td>
<td>c = a &amp; b; Sets bits in c to digital 1 if corresponding bits in both a and b are 1. For example, in binary notation, if a is 101000010 and b is 00000110, then c is set to 00000010.</td>
</tr>
<tr>
<td>~</td>
<td>Complement</td>
<td>c = ~ a; Inverts the bits. For example, in binary notation, if a is 101000010, then c is set to 010111101.</td>
</tr>
</tbody>
</table>
LOGICAL TESTS

<table>
<thead>
<tr>
<th>Test</th>
<th>Meaning</th>
<th>Example/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>If not false or zero</td>
<td>if (c); Subsequent command(s) will be processed if the test is true – for example, that c is a value; that is not 0.</td>
</tr>
<tr>
<td>!</td>
<td>Opposite (NOT)</td>
<td>if !(c); Subsequent command(s) will be processed if the opposite is true – for example, if c does not have value.</td>
</tr>
<tr>
<td>==</td>
<td>Tests equality</td>
<td>if (a == 4); Subsequent command(s) will be processed only if the test is true – that a equals 4, in the example.</td>
</tr>
<tr>
<td>!=</td>
<td>Test inequality</td>
<td>if (a != 4); Subsequent command(s) will be processed if the test is true, that a does not equal 4, in the example.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>if (a &gt; 4); Subsequent command(s) will be processed if the test is true, that a is greater than 4, in the example.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>if (a &lt; 4); Subsequent command(s) will be processed only if the test is true, that a is less than 4, in the example.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>if (a &gt;= 4); Subsequent command(s) will be processed if it is true that a is greater than or equal to 4, in the example.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td>if (a &lt;= 3); Subsequent command(s) will be processed if it is true that a is less than or equal to 4, in the example.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>All tests true (AND)</td>
<td>if (a == 4) &amp;&amp; (b &gt;= c); Subsequent command(s) will be processed if both test are true – that both a is equal to 4 and that b is greater than or equal to c, in the example.</td>
</tr>
</tbody>
</table>

ROBOT MOTION/DRIVE COMMANDS

<table>
<thead>
<tr>
<th>Command</th>
<th>Example/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>move (int ±mm)</td>
<td>move (-4000); Moves the robot forward (+) or backward (-) the specified integer number of millimeters – backwards four meters, in the example. The robot drives at sfSetMaxVelocity maximum speed.</td>
</tr>
<tr>
<td>turn (int ±degrees)</td>
<td>turn(90); Turns the robot at sfSetMaxVelocity speed the integer (mod360) degrees around from its current heading. The robot turns in the direction of the least number of degrees 0-180. Otherwise, positive (+) degrees &lt;180 turn the robot counterclockwise; negative degrees -1 to -180 turn the robot clockwise. The example turns the robot 90 degrees counterclockwise. (Unintuitive, we know.)</td>
</tr>
<tr>
<td>tumto (int ±degrees)</td>
<td>tumto(-45); Turns the robot to an absolute heading at sfSetMaxRVelocity rotational speed. Headings range 360 degrees; values are subtracted counterclockwise from the start-up direction (0 degrees). Negative rotation from 360 to resolve an absolute heading. Robot always rotates in the shortest direction. If the robot is heading in a direction 180 degrees from its start-up direction (turned completely around), in the example, it will turn counterclockwise to a heading of 225 degrees.</td>
</tr>
</tbody>
</table>
### PROGRAM LOGIC, SYNTAX AND COMMANDS

<table>
<thead>
<tr>
<th>Type</th>
<th>Command &amp; Syntax</th>
<th>Example/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional</td>
<td>if (test) {body}</td>
<td>if (a &gt; 5) move (100); Performs subsequent command or bracketed sequence of commands (called the body) if the test is true. Otherwise, optional else single command or body, if included, continuing to the next command in the activity.</td>
</tr>
<tr>
<td>Conditional</td>
<td>while (test) {body}</td>
<td>while (a&gt;5) a = a – 1; Continue to perform a command or bracketed sequence of commands as long as the test is true.</td>
</tr>
<tr>
<td>Conditional</td>
<td>waitfor (test)</td>
<td>waitfor (sfIsConnected); Waits for some condition to become true or have value, not zero. The example waits for a robot connection.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>wait (int msec)</td>
<td>wait(30); Pauses the current activity for the integer milliseconds; 30, in the example.</td>
</tr>
<tr>
<td>Motion Command Modifier</td>
<td>command() noblock</td>
<td>move (300) noblock; Colbert's move and turn commands normally block subsequent command execution and wait until the robot reaches its destination before allowing the activity to continue. The command modifier noblock tells the activity to continue without waiting.</td>
</tr>
<tr>
<td>Motion Command Modifier</td>
<td>command() timeout int cycles</td>
<td>move (4000) timeout 10; The Colbert motion command modifier timeout limits the amount of time move or turn will block subsequent execution of the activity. In the example, the robot has 10 cycles (approximately 1 second)* to move forward 4 meters. After that time, the robot may continue to move to the distance goal.</td>
</tr>
<tr>
<td>Activity Management Commands and Modifiers</td>
<td>act name (variable, ...) {body of activity}</td>
<td>act rectangle(int length, int width); Defines an activity labeled name and enclosed in the subsequent curly brackets (body). Optional variable declarations—separated by commas and enclosed in parentheses—let you set entry values for the activity. In the example, for instance, the square activity takes variable entry values for length and width that may be subsequently used by the activity commands to move the robot in a rectangle.</td>
</tr>
<tr>
<td>Activity Management</td>
<td>start name (value, value...) modifier</td>
<td>start square (100);</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Activity Management</td>
<td>trace name</td>
<td>trace;</td>
</tr>
<tr>
<td>Activity Management</td>
<td>notrace name</td>
<td>notrace square;</td>
</tr>
<tr>
<td>Activity Management</td>
<td>suspend name (cycles)</td>
<td>suspend square;</td>
</tr>
<tr>
<td>Activity Management</td>
<td>resume name</td>
<td>resume rectangle;</td>
</tr>
<tr>
<td>Activity Management</td>
<td>remove name</td>
<td>remove square;</td>
</tr>
</tbody>
</table>

* A cycle typically is 100 milliseconds long, but may vary with Colbert versions.
## ROBOT STATUS VARIABLES

Precede all variables with `sfRobot`. These variables are for reading position, velocities and so on; they cannot be used to set the position, velocity, etc. See Commands and Functions for that purpose.

<table>
<thead>
<tr>
<th>Variable Type &amp; Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float ax, ay, ath</td>
<td>while (sfRobot.ax&gt;sfRobot.ay) Cartesian X, Y (mm, mm) position and heading (degrees counterclockwise) relative to where and in which direction the robot was facing when first connected with Colbert. Example tests whether the x coordinate is larger than the y.</td>
</tr>
<tr>
<td>float dist</td>
<td>if (sfRobot.dist&gt;4000) Program-related distance traveled in mm, for example according to the last move command. Example will test whether the distance traveled was greater than 4000mm.</td>
</tr>
<tr>
<td>float tv, rv</td>
<td>if (sfRobot.tv&gt;sfSonarReading(3)) Current translational (mm/sec) and rotational (deg/sec) velocities, respectively. Example will test whether the robot's forward velocity is greater than its reading of sonar 3.</td>
</tr>
<tr>
<td>float leftv, rightv</td>
<td>while (sfRobot.leftv&gt;sfRobot.rightv) Current left and right wheel velocities, respectively, in mm/sec. Example tests whether the left velocity is greater than the right velocity.</td>
</tr>
<tr>
<td>float acc, macc, mtv, mrv</td>
<td>if (sfRobot.acc==sfRobot.macc) Current acceleration, maximum acceleration, maximum translational velocity, and maximum rotational velocity, respectively. Example will be true if the robot is traveling at maximum velocity.</td>
</tr>
<tr>
<td>int battery</td>
<td>sfSMessage (&quot;Charge: %d&quot;, sfRobot.battery/10) Battery voltage level in tenths of a volt. Example displays the remaining voltage, no tenths, after the text Charge:</td>
</tr>
<tr>
<td>int compass;</td>
<td>while (sfRobot.compass&lt;90) Compass heading 0-359, if installed. Example tests whether the compass reading is 90 or fewer degrees NE of North (0/360).</td>
</tr>
</tbody>
</table>
**PARTIAL LIST OF OTHER COLBERT FUNCTIONS**

Additional functions are described in the ActivMedia Software Guide and the Saphira User’s Guide.

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int sfStalledMotor(int which)</td>
<td>Returns 1 if motor(which) is stalled, otherwise 0; which motors are sfLEFT (0) and sfRIGHT (1)</td>
</tr>
<tr>
<td>int sfSetVelocity (int speed)</td>
<td>Sets the velocity of the robot to speed mm/sec. If negative, the robot moves backwards</td>
</tr>
<tr>
<td>void sfSetRVelocity(int speed)</td>
<td>Rotates the robot at speed degrees/sec. If negative, the robot moves clockwise.</td>
</tr>
<tr>
<td>void sfSetVelocity2(int left speed, int right speed)</td>
<td>Sets individual wheel velocities to respective speeds in mm/sec. If negative, the wheel moves backwards.</td>
</tr>
<tr>
<td>void sfSetMaxVelocity(int speed)</td>
<td>Sets the maximum velocity of the robot to speed mm/sec. Position commands will use this as their maximum; velocity commands will not be allowed to exceed it</td>
</tr>
<tr>
<td>void sfSetMaxRVelocity(int speed)</td>
<td>Sets the maximum rotational velocity of the robot to speed degrees/sec. Turn commands will use this as their maximum; rotate commands will not be allowed to exceed it</td>
</tr>
<tr>
<td>void sfSetPosition(int distance)</td>
<td>Moves the robot forward (positive) or backwards (negative) the distance in mm. Check for completion with sfDonePosition.</td>
</tr>
<tr>
<td>void sfIncHeading(int dhead)</td>
<td>A direct action turning the robot dhead degrees from its current heading. This differs from sfSetDCHeading, which turns differentially from the current control point. dhead may be in the interval [-180,180]. Negative angles are clockwise. Can be checked for completion with sfDoneHeading()</td>
</tr>
<tr>
<td>void sfSetHeading(int head)</td>
<td>A direct action turning the robot to head degrees in its internal coordinate system. head may be in the interval [-360,360]. Negative angles are clockwise. Can be checked for completion with sfDoneHeading()</td>
</tr>
<tr>
<td>int sfDonePosition(int delta)</td>
<td>Returns 1 if a direct position movement (sfSetPosition) is within delta mm of achieving its goal</td>
</tr>
<tr>
<td>int sfDonePosition(int delta)</td>
<td>Returns 1 if a direct heading movement (sfSetHeading, sfIncHeading) is within delta deg of achieving its goal</td>
</tr>
</tbody>
</table>
**int sfDonePosition(int delta)**  
Returns 1 if a direct heading movement (sfSetHeading, sfIncHeading) is within `delta` deg of achieving its goal.

**int sfOccPlane(int xy, int source, int d, int s1, int s2)**  
Returns the distance to the nearest sonar reading in the given subplane if there is one, and 5000 if not. The subplane is defined starting distance `d` from the robot center, with sides at `s1` and `s2`.  
`xy` specifies the orientation of the plane: sfFRONT, sfBACK, sfLEFT, or sfRIGHT. The source is one of sfFRONT (front sonars only), sfSIDES (side sonars only), or sfALL.

**int sfOccPlaneRet(int xy, int source, int d, int s1, int s2, float *x, float *y)**  
Returns the distance to the nearest sonar reading in the given subplane if there is one, and 5000 if not. Subplane is defined starting a distance `d` from the robot center, with sides at `s1` and `s2`. `xy` specifies orientation of the plane: sfFRONT, sfBACK, sfLEFT, or sfRIGHT. The source is one of sfFRONT (front sonars only), sfSIDES (side sonars only), or sfALL. In addition, the `x` and `y` values of the nearest sonar point are returned in `x` and `y`.

**int sfOccBox(int xy, int cx, int cy, int h, int w)**  
Returns the closest distance to a sonar reading if there is one in the given rectangle, and 5000 if not. The rectangle is defined by its center point, height, and width. `xy` specifies the direction in which to judge the distance: sfFRONT or sfSIDES. Note that a reading on the left or right side is always returned as a positive distance.

**int sfOccBoxRet(int xy, int cx, int cy, int h, int w, float *x, float *y)**  
Returns the closest distance to a sonar reading if there is one in the given rectangle, and 5000 if not. The rectangle is defined by its center point, height, and width. `xy` specifies the direction in which to judge the distance: sfFRONT or sfSIDES. Note that a reading on the left or right side is always returned as a positive distance. In addition, the `x` any `y` values of the closest point are returned.

**void sfMessage(string fmt, ...)**  
Prints a string in the Colbert Interaction area.

**void sfSMessage(string fmt, ...)**  
Prints a formatted string in the interaction area. Floating point numbers will print properly.

**int sfIsConnected()**  
Returns 1 when the client is connected to a robot, 0 if not.

**SfPlaySound() (int sound number)**  
(AmigoBot only) Play sound x from the current .ams playlist.
APPENDIX C: ADDITIONAL ROBOT RESOURCES
(also see Educator Support, page 3)

18. SRI's Artificial Intelligence Center, Saphira v.6 software with Colbert, ActivMedia Robotics, Peterborough, NH, 2000
19. Teacher's Corner PenPal Requests, www.theteacherscorner.net/penpals/index.htm, to find penpals for your classroom
INDEX

Activmedia Robotics, 6
Colbert Commands, 40, 62
Customizing Lessons, 2
Deliberative, 48
Demos, Colbert
  Demo 0, 41
  Demo 1, 44
  Demo 2, 45
  Demo 3, 48
  Demo 4, 52
  Demo 5, 54
  Demo 6, 42
Echolocation, 14
Effectors, 12
Equipment Requirements, 5
Foreign Language Instruction, 29
Integer, 44
IP Address: Finding, 27
Local Area Network (LAN), 25
Localize, 20
Loops, 45
Mapped Position, 20
Mapping
  Tips, 21
  Troubleshooting, 22
Microcontroller, 10
Motor Encoders, 14
Placement of Robots, 3
Programming, 39-60
Proprioception, 52
Reactive, 48
Remote Handling & Sensing, 36
Robot Control, 39
Scope And Sequence, 1
Sensed Position, 20
Set-Up
  AmigoBot, 4
  Other Robots, 4
  Internet, 5
  Simulator, 5
Social Science Instruction, 29
Sonar, 14
Status/States, 52
Support, 3
Troubleshooting Maps, 22
Uses of Robots In Education, 1
Internet Use, 5
Variable, 44