Teaching ROBOTC©
for LEGO© MINDSTORMS©

Product Overview

Robomatter INCORPORATED

Carnegie Mellon Robotics Academy

Why Robotics/STEM Education

We can't predict what the hot new technology will be in five years, but we can confidently predict that it will include computer programming, electronic embedded systems, engineering design, and mathematics. If you believe these things, then you need to know that robotics has the ability to teach these concepts. At the same time, robotics teaches 21st century skill sets like time management, resource allocation, teamwork, problem solving, and communications.

Think about this...

- Approximately 98% of all the 32-bit microprocessors currently in use worldwide are used in embedded systems; in other words they are being used in robotic smart technologies.
- By the year 2010, it is forecasted that 90% of the overall program code developed will be for embedded computing systems; to innovate and compete globally we will need more people that know how to program.
- Robotics Technology is a hundred billion dollar emerging industry that has moved from being an industry that could potentially employ thousands of people to an *integral part of all industries*. Robotics will impact the economy the same way that mass production impacted the industrial revolution and the computer impacted the information age.
- Science and Engineering (S&E) occupations are projected to grow by 26% from 2010 to 2020, twice as fast as the overall job market during that period (S&EI 2008) yet we have fewer students pursuing S&E careers.

The Teaching ROBOTC for LEGO MINDSTORMS Curriculum

If you are reading this, then you are probably considering teaching using the NXT and ROBOTC. I understand the dilemma that you are facing, I taught in the Pittsburgh Public School System for 27 years before coming to Carnegie Mellon. You have a limited budget, you have a thousand things to do every day that don't involve teaching, and then you also teach five or six sections of students. (Sometimes with multiple preps!) This curriculum is designed for a teacher with no programming background that is interested in teaching programming and engineering. This curriculum also supports a teacher that knows how to program, but has students of various skill levels and wants to allow them to move at their own pace. The teaching materials that we've developed use a high level of multimedia and have been tested with hundreds of students; they work. ROBOTC is the best programming software available for use with the NXT if you consider the percentage of teams that use ROBOTC software and made it to the finals of the FTC competition. You have one of the hardest jobs in the world; you teach. This training tool will make it easier.

Math is the language of science, engineering and technology

Many teachers see robotics as a way to teach STEM education. We've seen that robotics *does* provide unique opportunities for teachers to place engineering design, scientific process, technological literacy and mathematics in contexts that students find engaging and understand. Across the nation, many schools and community-based organizations are using robotics to address STEM competencies. Yet, our research is finding that many teachers miss key STEM "teaching moments" that robotics enables. Often, robotics teachers will allow students to be haphazard in their design process and avoid mathematics when possible (e.g., using guess-and-check strategies). This methodology leads to weak solutions and reduces student learning.

At the Robotics Academy we believe that:

- Math is the language of STEM and if you can't do math, then you won't be able to compete for a STEM job.
- Mathematics needs to be carefully thought out by the teacher and foregrounded for the student. The focus of the math instruction must be centered on addressing specific mathematics concepts (not general) and the mathematics in the lesson must be made explicit not implicit.
- For students' STEM understanding to move beyond parroting the teacher's words, ideas, and solutions, and to develop deep understanding, students need the opportunity to struggle with the problem, be able to defend their decisions, and explain their answer in their own words.

The moral of the story is "require your students to do the math." To learn more about teaching with robotics visit our web site. Carnegie Mellon is committed to helping teachers teach robotics. If there is something that we can help you with, then please contact us. If you see opportunities to make our teaching products stronger, then please contact us.

Have a great year.

Robin Shoop,
Director, Carnegie Mellon Robotics Academy

© 2009 Carnegie Mellon Robotics Academy

Teaching ROBOTC for LEGO MINDSTORMS

Teaching is a craft and every teacher does it differently. This curriculum is designed to teach "engineering process" and "programming". The Robotics Academy has developed this curriculum to help teachers to teach and students to learn those competencies. The Robotics Academy is committed to helping teachers use robotics to teach science, technology, engineering, and mathematics. Teachers can find additional NXT resources to teach robotics and engineering at our NXT site, which is continually upgraded by the Robotics Academy, go to www.nxteacher.com.

The scope and sequence below is designed to help you to quickly find resources to teach your class. The number of days at the right will depend on the number of challenges and labs that your students complete.

Welcome to Robotics Class.....**1-2 days**

What are the goals of this class?
What is the definition of a robot? - SPA handout - page 37
What is the definition of engineering? - Videos and handouts - page 10 & 11
What is the definition of programming? - Videos and handouts - page 15
What is the definition of a system?
What does it mean to manage a project? - Videos and handouts - page 10

Class organization rules.....**1-2 days**

Grading/Rubrics for Evaluation – page 13
Lab Procedures – pages 13 & 14
Keeping an Engineering Journal – page 10

Safety.....**2-3 days and then ongoing**

General Safety handout – page 8
Safety Checklist handout – page 9
Safety Quiz – page 9

Introduction to the NXT Hardware.....**2-3 days then ongoing**

The NXT Controller – page 18
NXT Sensors – page 17
NXT Parts Identification – page 17
Building your first robot – page 21

Introduction to ROBOTC Software.....**1 days then ongoing**

Download Firmware video – page 21
Download Your First Program lesson video – page 21
Introduction to ROBOTC 2.0 Software lesson video – page 15
Programming Quizzes and handouts – page 22

Introduction to Programming.....**1-2 days**

Thinking About Programming lesson video – page 15
ROBOTC Programming Syntax lesson video – page 16
Behaviors/psuedocode handout – page 15
Whitespace/Comments/ reserved words handouts – page 16

Movement – NXT Forward/Backward/Turning.....**8-10 days**

Labyrinth Challenge – page 27
Moving Forward lesson videos and handouts – pages 28 & 29
Motor Power Engineering Lab – page 30
Speed and Direction lesson videos and handouts – pages 29 & 30
Turning Engineering Lab – page 30
PID videos and handout – page 31
Synchronized Motors lesson video and handout – page 32
Synchronized Motors Engineering Lab – page 32
Introduction to Encoders lesson video and handouts – page 32

© 2009 Carnegie Mellon Robotics Academy

Teaching ROBOTC for LEGO MINDSTORMS

Sensing.....**20-25 days**
The Obstacle Course Programming Challenge – page 34

Touch Sensor – pages 35 through 37.....**3-5 days**
While Loop lesson video – page 36
While Loop reference handout – page 37
Sense-Plan-Act Algorithm reference handout – page 37
Boolean Logic lesson videos and handouts – pages 36 & 37
Touch Sensor Programming Challenges – page 37
Touch Sensor Quiz – page 37

Ultrasonic Sensor – pages 38 & 39.....**2-3 days**
Ultrasonic lesson video – page 39.
Calculating Thresholds handout – page 39
Random Numbers reference handout – page 39
Ultrasonic Programming Challenges – page 39
Ultrasonic Sensor Quiz – page 39

Encoders/the LEGO Smart Motors – pages 40 & 41.....**3-5 days**
Encoder lesson videos – page 41
Encoder Engineering Lab – page 41
Motor Encoder reference handout – page 41
Encoder programming Challenge – page 41

Light Sensor – pages 42 & 43.....**3-5 days**
Light Sensor lesson videos – page 43
Light Sensor Challenges – page 43
Light Sensor Quiz – page 43

Light Sensor/Line Tracking – pages 44 through 46.....**3-5 days**
Line Tracking lesson videos – page 45
Timer video – page 45
If-else Statement reference handout – page 46
Switch Case reference handout – page 46
Line Tracking Programming Challenges – page 46
Line Tracking Quiz – page 46

Sound Sensor – pages 47.....**2-3 days**
Sound Sensor lesson videos – page 47
Sound Sensor reference handout – page 47
Sound Sensor Quiz – page 47

Variables and Functions – page 48.....**10-15 days**
The Warehouse Programming Challenge – page 49

Automatic Threshold Calculations – pages 50 through 52.....**3-5 days**
Automatic Threshold lesson videos – page 51
Values and Variables lesson videos – page 51
Variables and the Debugger lesson video – page 51
Variables reference handout – page 52
Text to Display lab – page 52
Automatic Calculations Programming Challenge – page 52
Automatic Thresholds Quiz – page 52

Variables and Functions/Counting – pages 53 through 55.....**3-5 days**
Line Counting lesson videos – page 54
Quick Tap Programming Challenges – page 55
Line Counting Quiz – page 55

© 2009 Carnegie Mellon Robotics Academy

Teaching ROBOTC for LEGO MINDSTORMS

Variables and Functions/Patterns of Behaviors- pages 56 through 58.....**3-5 days**
Variables and Functions lesson videos – page 57
Global Variables reference handout – page 58
Functions reference handout – page 58
Functions Programming Challenges – page 58
Functions Programming Quiz – page 58

Debugging – page 59.....**2 days**
Debugging lesson videos – page 59

Remote Control – page 60.....**4-10 days**
Remote Control Soccer Programming Challenge – page 61

Remote Control Basics – pages 62 & 63
Remote Control lesson videos – page 63
Remote Control reference handout – page 63
Remote Control Buttons lab – page 63
Remote Control Programming Challenge – page 63

Using Bluetooth - pages 64 & 65
Using Bluetooth lesson videos – page 65
USB Bluetooth Adaptor reference handout page 65

How Robotics Addresses Standards - pages 76 through 85
National Science Education Standards - pages 76 through 79
National Council of Teachers of Mathematics - pages 80 & 81
International Technology Education Association - pages 82 & 83
Reading, Writing, Listening, and Presenting - pages 84 & 85

Additional Robotics Academy LEGO Robotics Resources - pages 86 & 87

© 2009 Carnegie Mellon Robotics Academy

Getting Started

The Teaching ROBOTC for LEGO MINDSTORMS is a comprehensive guide that teaches how to program the NXT Mindstorms hardware system as it helps student develop engineering competencies.

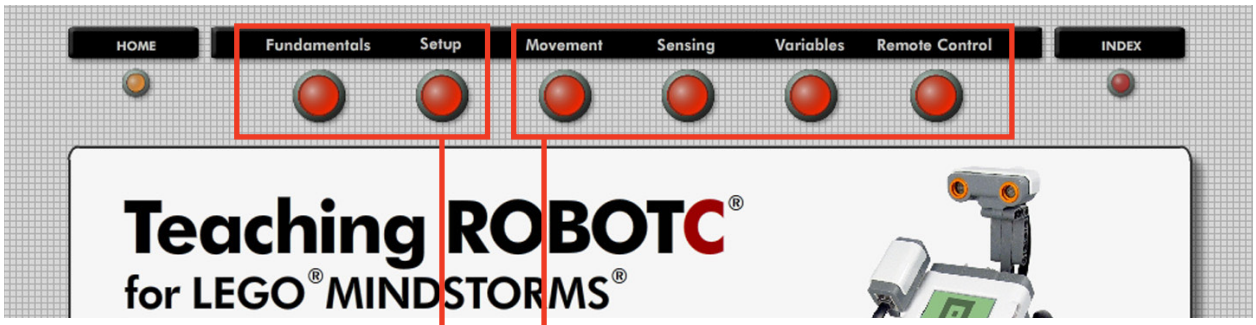
Scaffolded Approach to Learning

This curriculum uses a scaffolded approach to teach students how to program the NXT system. Students begin by learning to use, troubleshoot, and program the LEGO MINDSTORMS NXT robot system while advancing to more complex programming concepts. Once students understand how to program and troubleshoot the NXT system, there are Programming Challenges, Mini-challenges, and Engineering Challenges to practice what youv'e learned.

© 2009 Carnegie Mellon Robotics Academy

Getting Started/Fundamentals

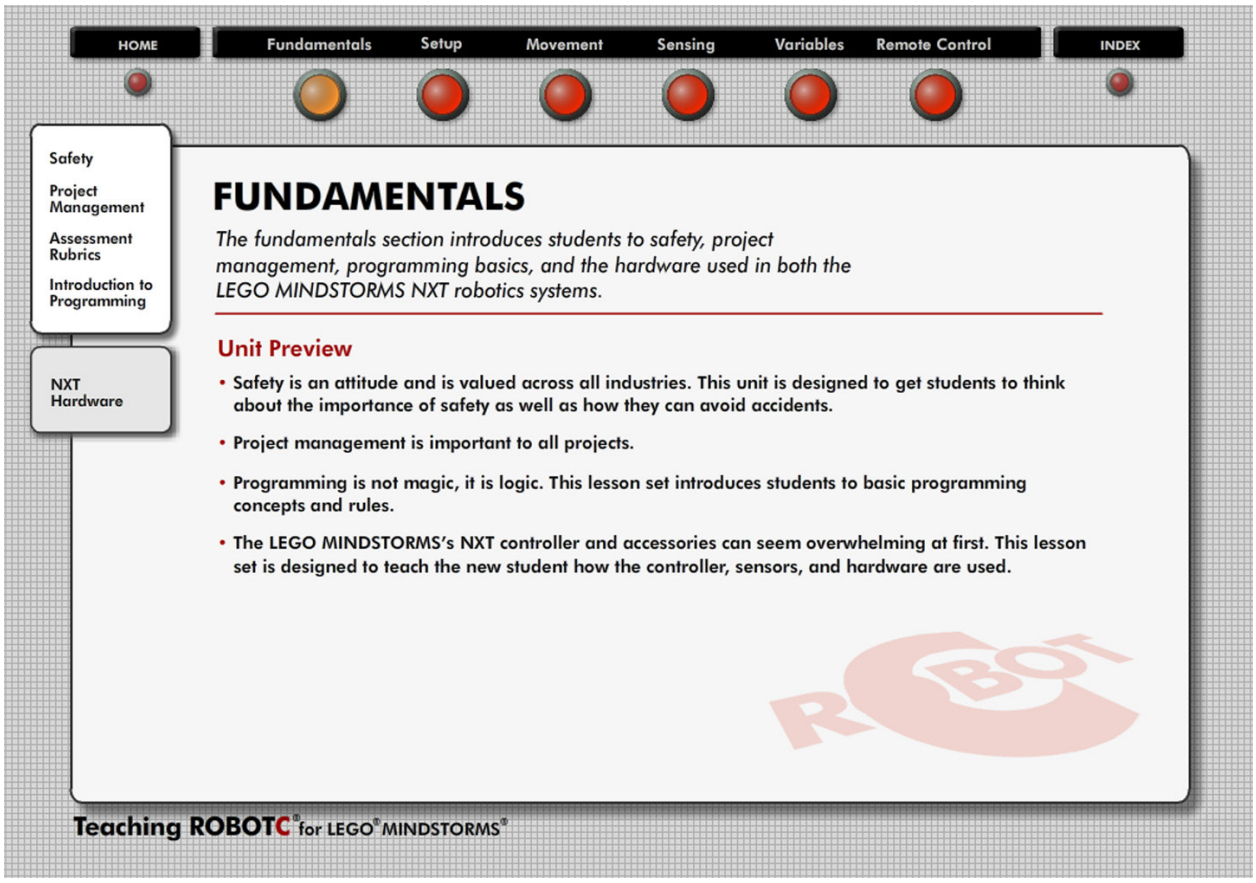
The curriculum is divided into two sections: “Getting Started” and “Programming and Engineering.”



The ***Getting Started Lessons*** are found in the “Fundamentals” and “Setup” sections.

The ***Programming and Engineering Lessons*** are found in the Movement, Sensing, Variables, and Remote Control” sections.

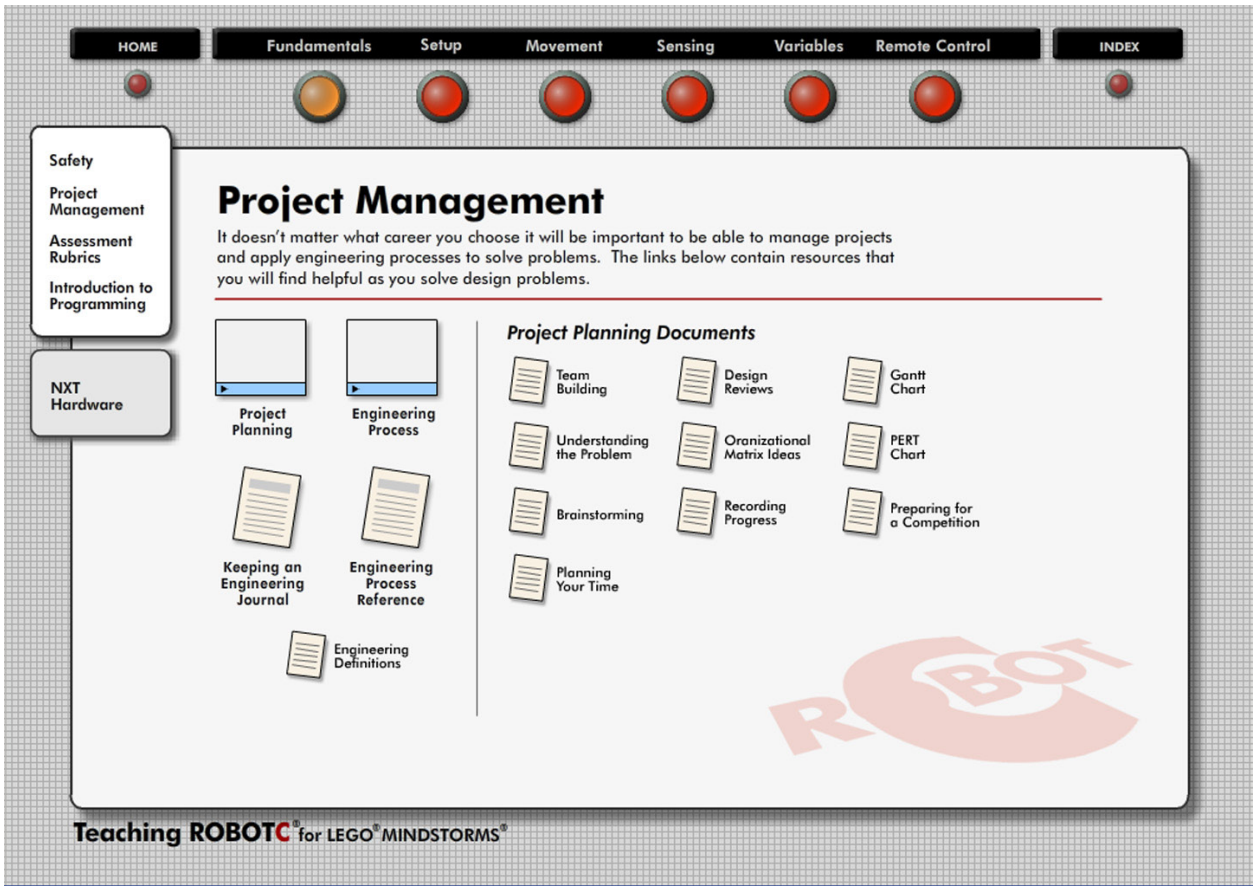
The Fundamentals Unit



The Fundamentals Unit is divided into six Lesson Sets: ***Safety, Project Management, Assessment Rubrics, Introduction to Programming, and NXT hardware***.

© 2009 Carnegie Mellon Robotics Academy

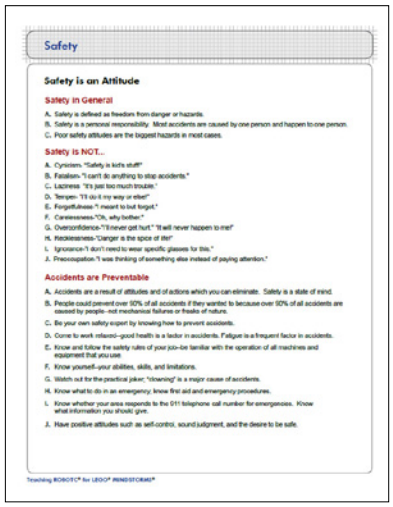
Getting Started/Fundamentals/Safety



Safety

Any course that involves moving parts, handling and processing materials and students requires safety training. Safety begins with the development of a safe attitude. Most accidents can be avoided if a student develops a safe and conscientious attitude. The safety lesson begins by challenging a student's general beliefs about safety and concludes with a safety inspection of the robotics lab.

Safety is an Attitude - A one page handout that defines what safety is and what safety is not, and concludes with statements that support the fact that most accidents are preventable with the development of a safe attitude.


Safety

Safety is an Attitude

Safety in General

A. Safety is defined as freedom from danger or hazards.
B. Safety is a personal responsibility. Most accidents are caused by one person and happen to one person.
C. Poor safety attitudes are the biggest hazards in most cases.

Safety is NOT...

A. Cynicism-"Safety is kid's stuff!"
B. Fatalism-"I can't do anything to stop accidents."
C. Laziness-"It's just too much trouble."
D. Temper-"I'll do it my way or else!"
E. Forgetfulness-"I meant to but forgot."
F. Carelessness-"Oh, why bother."
G. Overconfidence-"I'll never get hurt." "It will never happen to me!"
H. Recklessness-"Danger is the spice of life!"
I. Ignorance-"I don't need to wear specific glasses for this."
J. Preoccupation-"I was thinking of something else instead of paying attention."

Accidents are Preventable

A. Accidents are a result of attitudes and of actions which you can eliminate. Safety is a state of mind.
B. People could prevent over 90% of all accidents if they wanted to because over 90% of all accidents are caused by people--not mechanical failures or freaks of nature.
C. Be your own safety expert by knowing how to prevent accidents.
D. Come to work relaxed--good health is a factor in accidents. Fatigue is a frequent factor in accidents.
E. Know and follow the safety rules of your job-be familiar with the operation of all machines and equipment that you use.
F. Know yourself--your abilities, skills, and limitations.
G. Watch out for the practical joker; "clowning" is a major cause of accidents.
H. Know what to do in an emergency; know first aid and emergency procedures.
I. Know whether your area responds to the 911 telephone call number for emergencies. Know what information you should give.
J. Have positive attitudes such as self-control, sound judgment, and the desire to be safe.

Teaching ROBOTC® for LEGO® MINDSTORMS®

General Lab Safety - A four page handout that spells out general safety rules, describes features of a safe classroom, safe storage, material handling, disposal of materials, tools and equipment, and ends with a list of definitions of terms that students may not know.

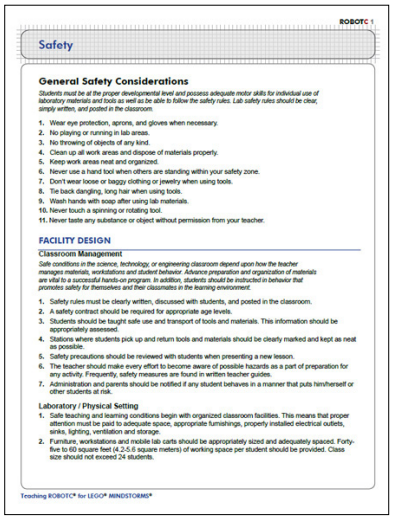
Safety Checklist - A three page handout that contains a safety checklist, rules to consider when you are moving things around the lab, and a one page safety poster.

Electrical Safety - A two page handout that describes safety rules when working with electricity and common causes of electrical accidents, including defective equipment, unsafe practices, and lack of electrical knowledge.

Power Tool Safety - A one page handout that sets rules and expectations for when students use power tools in the robotics lab.

© 2009 Carnegie Mellon Robotics Academy

Getting Started/Fundamentals/Safety continued



ROBOTC 1

Safety

General Safety Considerations

Students must be at the proper developmental level and possess adequate motor skills for individual use of laboratory materials and tools as well as be able to follow the safety rules. Lab safety rules should be clear, simply written, and posted in the classroom.

1. Wear eye protection, aprons, and gloves when necessary.
2. No playing or running in lab areas.
3. No throwing of objects of any kind.
4. Clean up all work areas and dispose of materials properly.
5. Keep work areas neat and organized.
6. Never use a hand tool when others are standing within your safety zone.
7. Don't wear loose or baggy clothing or jewelry when using tools.
8. Tie back dangling, long hair when using tools.
9. Wash hands with soap after using lab materials.
10. Never touch a spinning or rotating tool.
11. Never taste any substance or object without permission from your teacher.

FACILITY DESIGN

Classroom Management

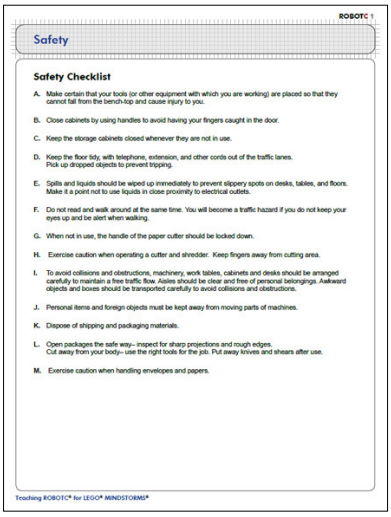
Safe conditions in the science, technology, or engineering classroom depend upon how the teacher manages materials, workstations and student behavior. Advance preparation and organization of materials are vital to a successful hands-on program. In addition, students should be instructed in behavior that promotes safety for themselves and their classmates in the learning environment.

1. Safety rules must be clearly written, discussed with students, and posted in the classroom.
2. A safety contract should be required for appropriate age levels.
3. Students should be taught safe use and transport of tools and materials. This information should be appropriately assessed.
4. Stations where students pick up and return tools and materials should be clearly marked and kept as neat as possible.
5. Safety precautions should be reviewed with students when presenting a new lesson.
6. The teacher should make every effort to become aware of possible hazards as a part of preparation for any activity. Frequently, safety measures are found in written teacher guides.
7. Administration and parents should be notified if any student behaves in a manner that puts him/herself or other students at risk.

Laboratory / Physical Setting

1. Safe teaching and learning conditions begin with organized classroom facilities. This means that proper attention must be paid to adequate space, appropriate furnishings, properly installed electrical outlets, sinks, lighting, ventilation and storage.
2. Furniture, workstations and mobile lab carts should be appropriately sized and adequately spaced. Forty-five to 60 square feet (4.2-5.6 square meters) of working space per student should be provided. Class size should not exceed 24 students.

Teaching ROBOTC® for LEGO® MINDSTORMS®



ROBOTC 1

Safety

Safety Checklist

A. Make certain that your tools (or other equipment with which you are working) are placed so that they cannot fall from the bench-top and cause injury to you.

B. Close cabinets by using handles to avoid having your fingers caught in the door.

C. Keep the storage cabinets closed whenever they are not in use.

D. Keep the floor tidy, with telephone, extension, and other cords out of the traffic lanes. Pick up dropped objects to prevent tripping.

E. Spills and liquids should be wiped up immediately to prevent slippery spots on desks, tables, and floors. Make it a point not to use liquids in close proximity to electrical outlets.

F. Do not read and walk around at the same time. You will become a traffic hazard if you do not keep your eyes up and be alert when walking.

G. When not in use, the handle of the paper cutter should be locked down.

H. Exercise caution when operating a cutter and shredder. Keep fingers away from cutting area.

I. To avoid collisions and obstructions, machinery, work tables, cabinets and desks should be arranged carefully to maintain a free traffic flow. Aisles should be clear and free of personal belongings. Awkward objects and boxes should be transported carefully to avoid collisions and obstructions.

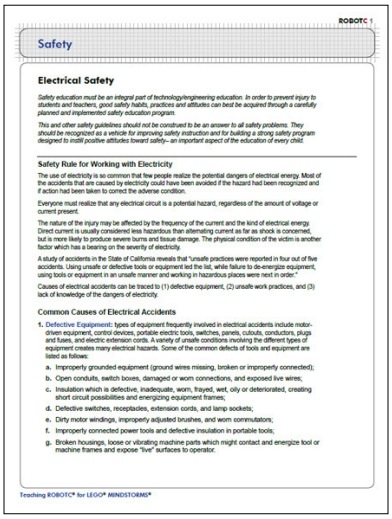
J. Personal items and foreign objects must be kept away from moving parts of machines.

K. Dispose of shipping and packaging materials.

L. Open packages the safe way– inspect for sharp projections and rough edges. Cut away from your body– use the right tools for the job. Put away knives and shears after use.

M. Exercise caution when handling envelopes and papers.

Teaching ROBOTC® for LEGO® MINDSTORMS®



ROBOTC 1

Safety

Electrical Safety

Safety education must be an integral part of technology/engineering education. In order to prevent injury to students and teachers, good safety habits, practices and attitudes can best be acquired through a carefully planned and implemented safety education program.

This and other safety guidelines should not be construed to be an answer to all safety problems. They should be recognized as a vehicle for improving safety instruction and for building a strong safety program designed to instill positive attitudes toward safety– an important aspect of the education of every child.

Safety Rule for Working with Electricity

The use of electricity is so common that few people realize the potential dangers of electrical energy. Most of the accidents that are caused by electricity could have been avoided if the hazard had been recognized and if action had been taken to correct the adverse condition.

Everyone must realize that any electrical circuit is a potential hazard, regardless of the amount of voltage or current present.

The nature of the injury may be affected by the frequency of the current and the kind of electrical energy. Direct current is usually considered less hazardous than alternating current as far as shock is concerned, but is more likely to produce severe burns and tissue damage. The physical condition of the victim is another factor which has a bearing on the severity of electricity.

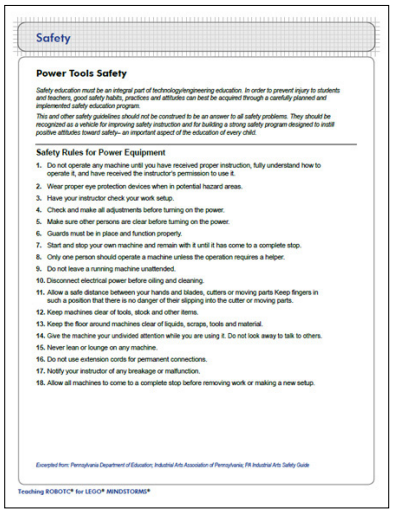
A study of accidents in the State of California reveals that "unsafe practices were reported in four out of five accidents. Using unsafe or defective tools or equipment led the list, while failure to de-energize equipment, using tools or equipment in an unsafe manner and working in hazardous places were next in order."

Causes of electrical accidents can be traced to (1) defective equipment, (2) unsafe work practices, and (3) lack of knowledge of the dangers of electricity.

Common Causes of Electrical Accidents

1. Defective Equipment: types of equipment frequently involved in electrical accidents include motor-driven equipment, control devices, portable electric tools, switches, panels, cutouts, conductors, plugs and fuses, and electric extension cords. A variety of unsafe conditions involving the different types of equipment creates many electrical hazards. Some of the common defects of tools and equipment are listed as follows:
   a. Improperly grounded equipment (ground wires missing, broken or improperly connected);
   b. Open conduits, switch boxes, damaged or worn connections, and exposed live wires;
   c. Insulation which is defective, inadequate, worn, frayed, wet, oily or deteriorated, creating short circuit possibilities and energizing equipment frames;
   d. Defective switches, receptacles, extension cords, and lamp sockets;
   e. Dirty motor windings, improperly adjusted brushes, and worn commutators;
   f. Improperly connected power tools and defective insulation in portable tools;
   g. Broken housings, loose or vibrating machine parts which might contact and energize tool or machine frames and expose "live" surfaces to operator.

Teaching ROBOTC® for LEGO® MINDSTORMS®



Safety

Power Tools Safety

Safety education must be an integral part of technology/engineering education. In order to prevent injury to students and teachers, good safety habits, practices and attitudes can best be acquired through a carefully planned and implemented safety education program.

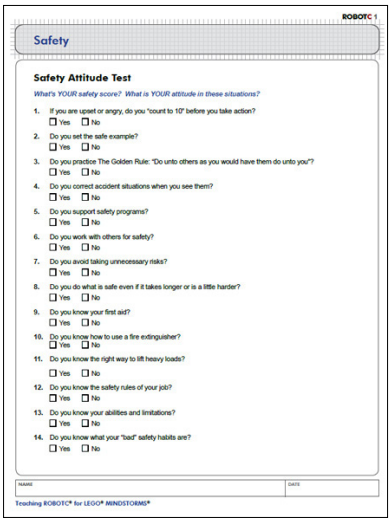
This and other safety guidelines should not be construed to be an answer to all safety problems. They should be recognized as a vehicle for improving safety instruction and for building a strong safety program designed to instill positive attitudes toward safety– an important aspect of the education of every child.

Safety Rules for Power Equipment

1. Do not operate any machine until you have received proper instruction, fully understand how to operate it, and have received the instructor's permission to use it.
2. Wear proper eye protection devices when in potential hazard areas.
3. Have your instructor check your work setup.
4. Check and make all adjustments before turning on the power.
5. Make sure other persons are clear before turning on the power.
6. Guards must be in place and function properly.
7. Start and stop your own machine and remain with it until it has come to a complete stop.
8. Only one person should operate a machine unless the operation requires a helper.
9. Do not leave a running machine unattended.
10. Disconnect electrical power before oiling and cleaning.
11. Allow a safe distance between your hands and blades, cutters or moving parts Keep fingers in such a position that there is no danger of their slipping into the cutter or moving parts.
12. Keep machines clear of tools, stock and other items.
13. Keep the floor around machines clear of liquids, scraps, tools and material.
14. Give the machine your undivided attention while you are using it. Do not look away to talk to others.
15. Never lean or lounge on any machine.
16. Do not use extension cords for permanent connections.
17. Notify your instructor of any breakage or malfunction.
18. Allow all machines to come to a complete stop before removing work or making a new setup.

Excerpted from: Pennsylvania Department of Education; Industrial Arts Association of Pennsylvania; PA Industrial Arts Safety Guide

Teaching ROBOTC® for LEGO® MINDSTORMS®



ROBOTC 1

Safety

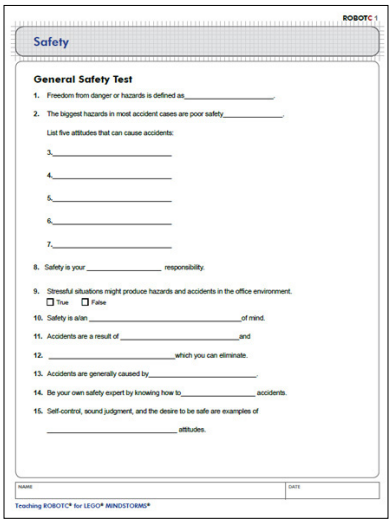
Safety Attitude Test

What's YOUR safety score? What is YOUR attitude in these situations?

1. If you are upset or angry, do you "count to 10" before you take action?
   ☐ Yes ☐ No
2. Do you set the safe example?
   ☐ Yes ☐ No
3. Do you practice The Golden Rule: "Do unto others as you would have them do unto you"?
   ☐ Yes ☐ No
4. Do you correct accident situations when you see them?
   ☐ Yes ☐ No
5. Do you support safety programs?
   ☐ Yes ☐ No
6. Do you work with others for safety?
   ☐ Yes ☐ No
7. Do you avoid taking unnecessary risks?
   ☐ Yes ☐ No
8. Do you do what is safe even if it takes longer or is a little harder?
   ☐ Yes ☐ No
9. Do you know your first aid?
   ☐ Yes ☐ No
10. Do you know how to use a fire extinguisher?
    ☐ Yes ☐ No
11. Do you know the right way to lift heavy loads?
    ☐ Yes ☐ No
12. Do you know the safety rules of your job?
    ☐ Yes ☐ No
13. Do you know your abilities and limitations?
    ☐ Yes ☐ No
14. Do you know what your "bad" safety habits are?
    ☐ Yes ☐ No

NAME	DATE

Teaching ROBOTC® for LEGO® MINDSTORMS®



ROBOTC 1

Safety

General Safety Test

1. Freedom from danger or hazards is defined as_____.
2. The biggest hazards in most accident cases are poor safety_____.

   List five attitudes that can cause accidents:

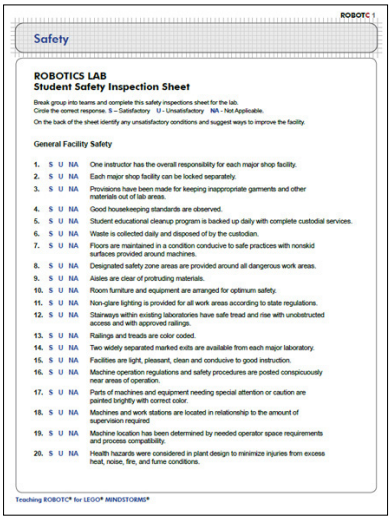
   3._____
   4._____
   5._____
   6._____
   7._____
8. Safety is your _____ responsibility.
9. Stressful situations might produce hazards and accidents in the office environment.
   ☐ True ☐ False
10. Safety is a/an _____of mind.
11. Accidents are a result of _____and
12. _____which you can eliminate.
13. Accidents are generally caused by_____.
14. Be your own safety expert by knowing how to_____ accidents.
15. Self-control, sound judgment, and the desire to be safe are examples of _____ attitudes.

NAME	DATE

Teaching ROBOTC® for LEGO® MINDSTORMS®

Safety Tests and Answers - Three different safety quizzes designed to check students' understanding of the importance of safety.

Robotics Lab Safety Inspection Sheet - Helps students to understand that they need to monitor the robotics classroom for safety.



ROBOTC 1

Safety

ROBOTICS LAB
Student Safety Inspection Sheet

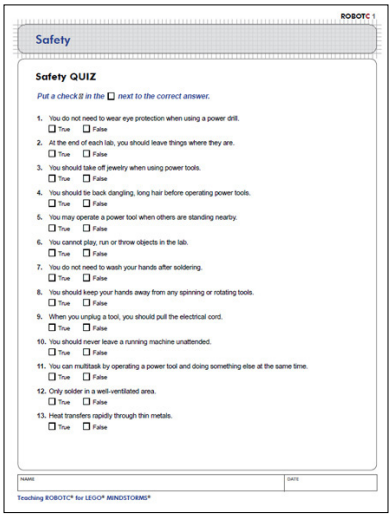
Break group into teams and complete this safety inspections sheet for the lab.
Circle the correct response. S – Satisfactory U - Unsatisfactory NA - Not Applicable.

On the back of the sheet identify any unsatisfactory conditions and suggest ways to improve the facility.

General Facility Safety

1. S U NA One instructor has the overall responsibility for each major shop facility.
2. S U NA Each major shop facility can be locked separately.
3. S U NA Provisions have been made for keeping inappropriate garments and other materials out of lab areas.
4. S U NA Good housekeeping standards are observed.
5. S U NA Student educational cleanup program is backed up daily with complete custodial services.
6. S U NA Waste is collected daily and disposed of by the custodian.
7. S U NA Floors are maintained in a condition conducive to safe practices with nonskid surfaces provided around machines.
8. S U NA Designated safety zone areas are provided around all dangerous work areas.
9. S U NA Aisles are clear of protruding materials.
10. S U NA Room furniture and equipment are arranged for optimum safety.
11. S U NA Non-glare lighting is provided for all work areas according to state regulations.
12. S U NA Stairways within existing laboratories have safe tread and rise with unobstructed access and with approved railings.
13. S U NA Railings and treads are color coded.
14. S U NA Two widely separated marked exits are available from each major laboratory.
15. S U NA Facilities are light, pleasant, clean and conducive to good instruction.
16. S U NA Machine operation regulations and safety procedures are posted conspicuously near areas of operation.
17. S U NA Parts of machines and equipment needing special attention or caution are painted brightly with correct color.
18. S U NA Machines and work stations are located in relationship to the amount of supervision required
19. S U NA Machine location has been determined by needed operator space requirements and process compatibility.
20. S U NA Health hazards were considered in plant design to minimize injuries from excess heat, noise, fire, and fume conditions.

Teaching ROBOTC® for LEGO® MINDSTORMS®



ROBOTC 1

Safety

Safety QUIZ

Put a check ll in the ☐ next to the correct answer.

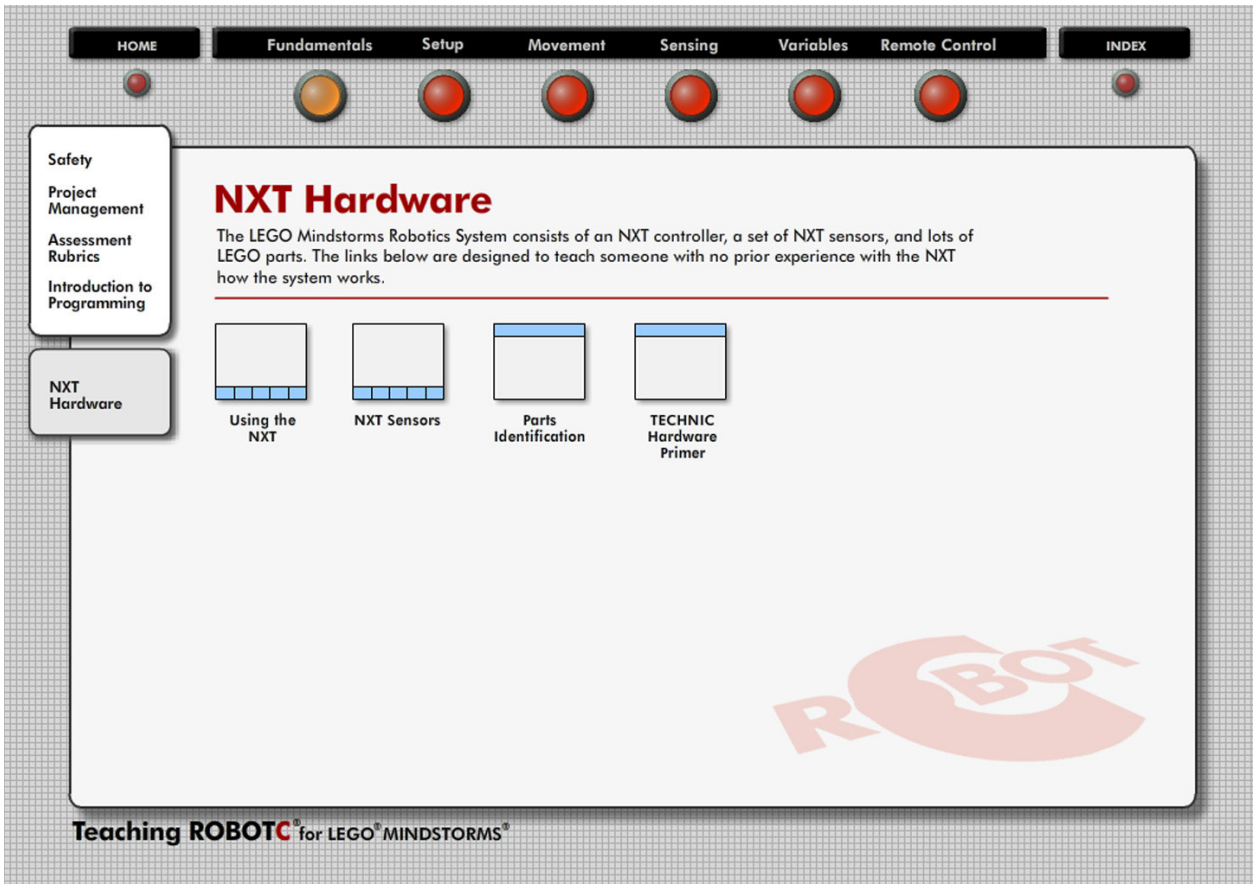
1. You do not need to wear eye protection when using a power drill.
   ☐ True ☐ False
2. At the end of each lab, you should leave things where they are.
   ☐ True ☐ False
3. You should take off jewelry when using power tools.
   ☐ True ☐ False
4. You should tie back dangling, long hair before operating power tools.
   ☐ True ☐ False
5. You may operate a power tool when others are standing nearby.
   ☐ True ☐ False
6. You cannot play, run or throw objects in the lab.
   ☐ True ☐ False
7. You do not need to wash your hands after soldering.
   ☐ True ☐ False
8. You should keep your hands away from any spinning or rotating tools.
   ☐ True ☐ False
9. When you unplug a tool, you should pull the electrical cord.
   ☐ True ☐ False
10. You should never leave a running machine unattended.
    ☐ True ☐ False
11. You can multitask by operating a power tool and doing something else at the same time.
    ☐ True ☐ False
12. Only solder in a well-ventilated area.
    ☐ True ☐ False
13. Heat transfers rapidly through thin metals.
    ☐ True ☐ False

NAME	DATE

Teaching ROBOTC® for LEGO® MINDSTORMS®

© 2009 Carnegie Mellon Robotics Academy

Getting Started/Fundamentals/Project Management



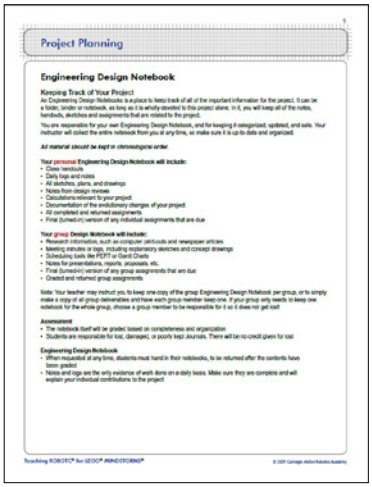
Project Management

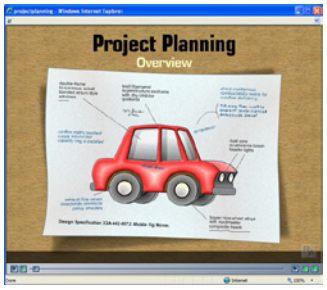
Many schools compete in robotic competitions; other schools are using MINDSTORMS to teach engineering. The links on this page provide students with resources that teach how to manage projects and solve engineering design problems.

Project Planning Video - The five minute Project Planning Video uses a combination of humor and examples to describe what a well planned project looks like.

Engineering Process Video - The five minute Engineering Process Video highlights the importance of research, planning, developing prototypes, and iterative testing when solving engineering design problems.

Engineering Design Notebook - Each student is required to keep an engineering design notebook. This two page handout describes the what is kept in the notebook and what a daily log is.





© 2009 Carnegie Mellon Robotics Academy

Getting Started/Fundamentals/Project Management continued

Project Planning Documents

Engineering Process Reference - This three page set of handouts describes steps that engineers use to solve problems, provides a set of definitions for the word “engineering”, and describes the iterative nature of design.

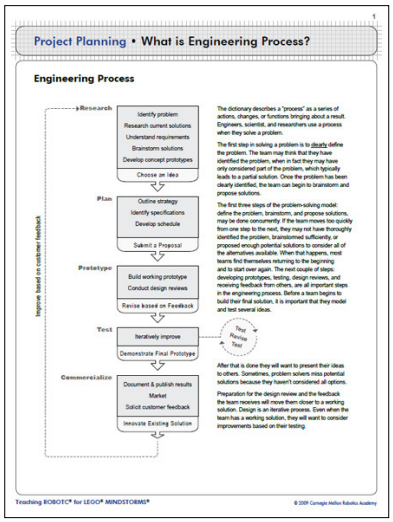
Team Building PDF- The Team Building Lesson Set consists of four documents that describe guidelines that teams should use for the first team meeting, general ground rules for any team meeting, things to consider when building teams, and a description of roles on a robotics team: Project Management, Programming, Engineering, Documentation and Communications.

Understanding the Problem PDF - One of the keys to solving any problem is “Understanding the Problem”. This three page set of handouts consists of: Defining the Problem, Technical Research, and Creating a Design Specification.

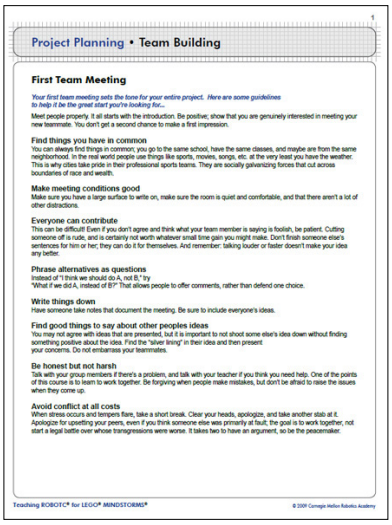
Brainstorming PDF - Another key problem-solving step involves meetings where people brainstorm together to develop potential solutions. This three page set of handouts is broken into: a Brainstorming Primer, Things to Think About, and Brainstorming Tips. Each handout can be used individually or as part of a set.

Planning Your Time PDF - Time management is a crucial skill to develop. This four page handout uses a simple activity, planning a birthday party, to describe a critical skill set that everyone should learn: how to manage time.

Design Reviews PDF - Engineers conduct design reviews on a regular basis. This two page handout describes how to conduct weekly team design reviews, as well as preliminary and detail design reviews.


Project Planning • What is Engineering Process?

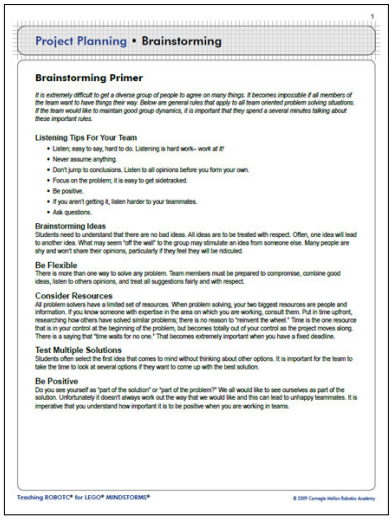
Engineering Process


Project Planning • Team Building

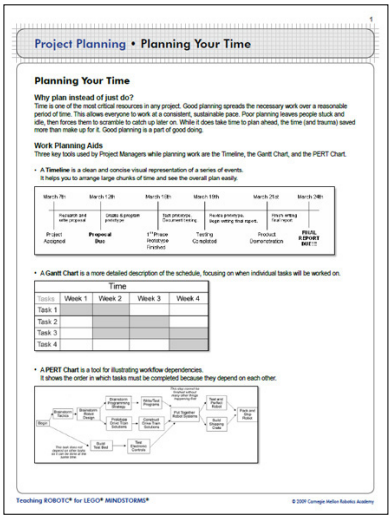
First Team Meeting

Project Planning • Understanding the Problem

Creating a Design Specification


Project Planning • Brainstorming

Brainstorming Primer


Project Planning • Planning Your Time

Planning Your Time

Project Planning • Design Reviews

The Design Review

© 2009 Carnegie Mellon Robotics Academy

Getting Started/Fundamentals/Project Management continued

Project Planning Documents

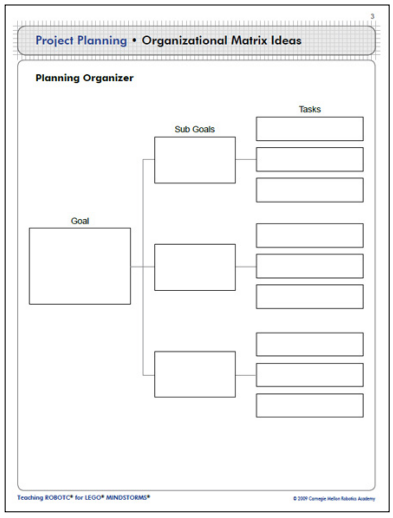
Organizational Matrix Ideas PDF - This three page handout graphically shows three methods of organizing projects.

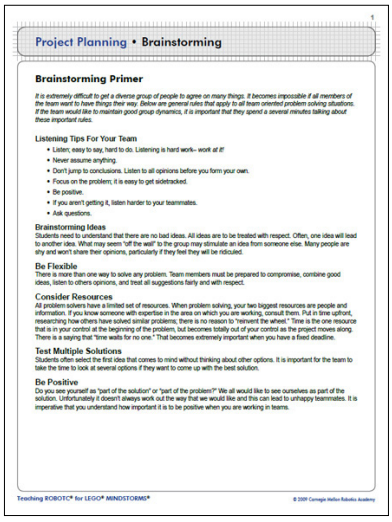
Recording Progress PDF - The recording progress tools offer the project manager three solutions that can be used to help team members to document the team's progress toward the project goals.

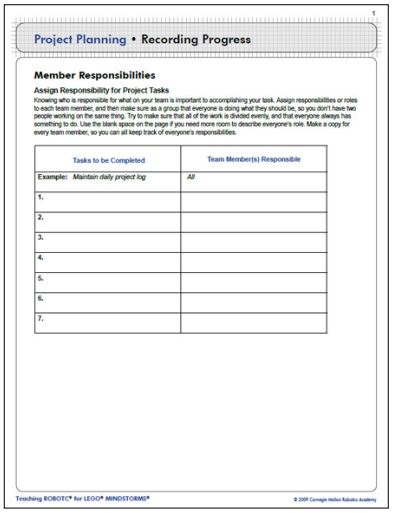
Gantt Chart PDF - A Gantt chart provides a graphical illustration of a schedule to help plan, coordinate, and track specific tasks in a project. This one page handout is designed to teach students how Gantt charts work.

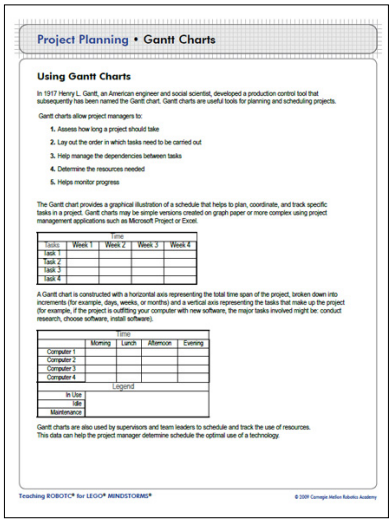
PERT Chart PDF - A PERT chart is a tool that graphically illustrates when parts of the project become due. The advantage of the PERT chart is that it shows which things must be completed in sequence and which things need to be completed simultaneously. This one page handout is designed to teach students how PERT charts work.

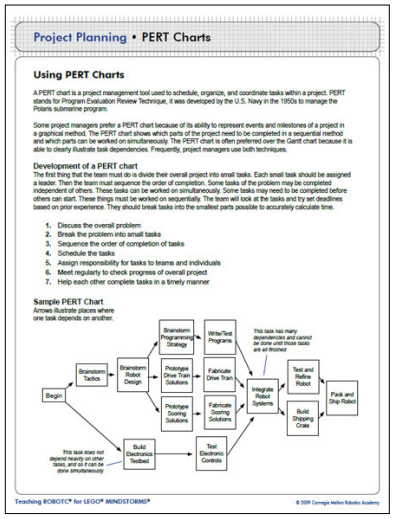
Preparing for a Competition PDF- Robotic competitions offer unique opportunities to teach students about time management, resource allocation, teamwork, and problem solving, all within a context that they find challenging but fun. The Preparing for a Competition handout is designed to support robotics teams as they plan for the competition.

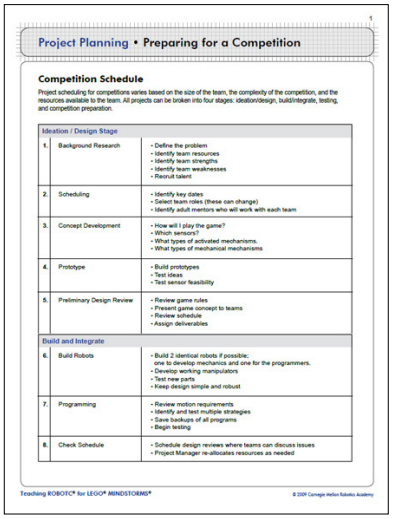






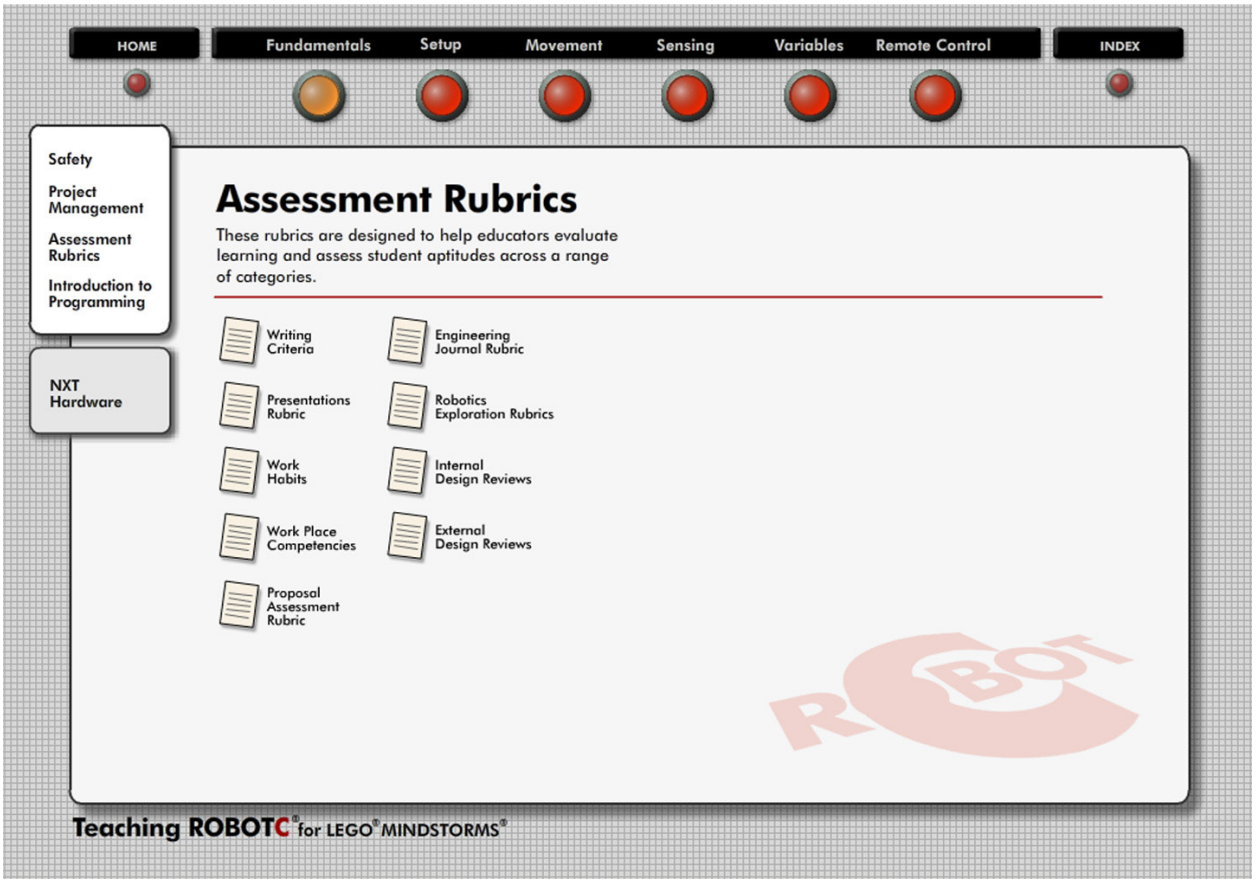






© 2009 Carnegie Mellon Robotics Academy

Getting Started/Fundamentals/Assessment Rubrics



Assessment Rubrics

Timely assessment is paramount in today's educational environment. A clear expectation of what is being assessed is a key to training students. Traditional assessments are provided in the curriculum; ie quizzes. The assessments found in this section are assessment rubrics for project-based learning. There are many other tools that a teacher may use, but this section provides some examples. Rubrics allow all stakeholders to see what is being measured.

Writing Criteria Rubric - Writing is a process and good writing requires several steps: brainstorming, outlining, prewriting, and editing. This is a simple rubric that check for those steps.

Engineering Journal Rubric - Explains to students what is expected in their engineering journals.

Presentation Rubric - Helps students determine what a good presentation should include.

Request for Proposal Rubric - Helps students to determine what is being evaluated in their RPF submission.

Work Habit Evaluation - This is a great tool for students to use to develop strong work habits.

Workplace Competencies Rubric - This rubric helps students to develop the skills that are valued by industry.

Internal Design Rubric - This evaluation tool helps students understand the expectations and preparation needed for an internal design review.

External Design Review - This evaluation tool helps students understand the expectations and preparation needed for an external design review.

© 2009 Carnegie Mellon Robotics Academy

Getting Started/Fundamentals/Assessment Rubrics continued

ROBOTC

Rubrics for Engineering Journal Assessment

The Engineering Journal

The Engineering Journal is a highly recommended organizational method for the instructor to keep track of each group's work throughout the multi-week project. It consists of a folder or binder for each individual in the class, which contains the entirety of that student's work for the project. Consolidating each student's work in a single place allows for easy collection of assignments, and gives students responsibility for keeping his or her own material organized.

This gives the instructor the option of collecting students' journals to grade when assignments are due (and even when they're not).

Each student's Engineering Journal contains:

- Class handouts
- Daily logs and notes
- All completed and returned assignments
- Final (turned-in) version of any individual assignments that are due

All material should be kept in chronological order

Alternatively, you may choose to have only one journal per group, or have every student keep a copy of both individual and group assignments. The Engineering Journal is your tool for efficient assessment – customize the requirements to fit the needs of your classroom.

Assessment

- The Journal itself should be graded based on completeness and organization
  - A complete journal should include:
    - All class handouts, including syllabus and assignment sheets
    - All teacher-assigned work (homework, quizzes, etc.)
    - Daily logs, one per day of independent work
    - All major project deliverables (proposal, etc. – group journal keeper only)
    - Group meeting notes (group journal keeper only)
  - All documents in the journal should be organized by date
- Students should be responsible for lost, damaged, or poorly kept Journals
  - Points should be deducted for journals that are:
    - Lost (no credit for assignments that are lost!)
    - Damaged or sloppy (unprofessional!)
- When requested, students should hand in their journals
  - This is the preferred method for collecting work on days assignments are due
    - Penalties apply for groups or individuals who are not prepared
- Journal contents should be graded and returned in the journal
  - Assignments should be graded according to their own rubrics
    - Quizzes and journal hand-ins can be done together for convenience
- Notes and logs are a student's evidence of work done on a daily basis
  - Self- and peer-reported student records are how work habits are tracked
    - Teamwork
    - Effective use of time
    - Good planning and preparation

ROBOTC® Curriculum for TETRIX™ and LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC

Rubrics for External Design Review

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Timeliness (10%)	• Prototype is fully functional at the time of review • Group is present and ready to begin on time	• Prototype is semi-functional (has at least one working component) at the time of review • Group is present and ready to begin on time	• Prototype is together, but not functional at the time of review • Group is present but not ready to present	• Prototype is not built or functional at time of review • Not all group members are present, and group is not ready
Presentation (15%)	• Discussion remains professional in tone and direction • Discussion proceeds efficiently due to understanding and articulation of the groups ideas and design • Group is able to focus on relevant aspects of the robot design	• Discussion remains professional in tone and direction • For the most part discussion proceeds efficiently • Group may not focus on relevant aspects of design, but does articulate important issues	• Discussion does not have a professional tone or manner • Discussion does not stay on topic • Students may articulate some ideas of the project, but do not focus on main aspects of robot design	• Little discussion occurs during the presentation • Students show lack of understanding of group ideas and design of robot • Group does not focus discussion
Project Management (30%)	• Development is in line with timeline submitted with proposal • Group member roles and responsibilities are defined and adhered to • Schedule for future development is practical and workable	• Development is mostly in line with timeline submitted with proposal • Group members have roles and responsibilities which were mostly adhered to • Schedule for future development is present and likely practical and workable	• Group member do not abide by timeline that was submitted with proposal • Group members do not adhere to clearly defined set of roles and responsibilities • Future development is present but may not be in schedule or practical and workable	• Timeline is not submitted, or timeline is disregarded during development process • Group members do not clearly define roles • No schedule for future development is created

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC

Rubrics for Internal Design Review

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Timeliness (10%)	• Design Candidate Sheets are completed on time for each design • Design Assessment Criteria sheet is completed on time • Group is present and ready to begin on time	• Design Candidate Sheets are complete but not on time • Design Assessment Criteria sheet is completed but not on time • Group is present and read to begin on time	• Most Design Candidate Sheets are complete • Design Assessment Criteria sheet is mostly complete • Group is present	• Design Candidate Sheets are not completed • Design Assessment Criteria sheet is not completed • Group is not present
Discussion (30%)	• Group follows good meeting and teamwork procedures • Discussion remains professional in tone and direction • Discussion proceeds efficiently • Group is able to focus on the relevant aspects of the robot designs	• Group follows decent meeting and teamwork procedures • Discussion remains professional in tone and direction • For the most part discussion proceeds efficiently • For the most part group focuses on relevant aspect of robot designs	• Group follows few meeting and teamwork procedures • Discussion does not have a professional tone or manner • Discussion does not proceed efficiently • Group rarely focuses on relevant aspects of robot design	• Group does not work as a team • Little discussion occurs • Discussion does not stay on topic • Group does not focus on relevant aspects of robot design

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC

Rubrics for Presentations

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Use of Multimedia Technology	Excellent use of multimedia technology. • The presentation was eye appealing. • The pictures were clear. • The sequence of the presentation was well thought out. • Presentation was organized. • Speakers were clear and used proper terminology. • Complete	Student demonstrated they knew how to use multimedia technology • The presentation was eye appealing. • The pictures were clear. • The sequence of the presentation was well thought out. • Complete	Multimedia technology use needs work. • The use of multimedia technology was a distraction rather than help. • Incomplete	Multimedia technology didn't support topic. • Pictures were not clear and didn't seem to have a purpose. • Incomplete
Content Analysis	Content was excellent. • Presentation was organized. • Project was fully described. • I learned something when I listened to the presentation. • Presentation enjoyable to watch.	Content was good. • Presentation was organized. • Presentation was organized but needed more practice to be excellent	Content of presentation lacked clarity. • Presentation lacked organization and didn't have a unified theme. • Presenters didn't use proper terminology.	Content of presentation lacked clarity. • Presentation lacked organization and didn't have a unified theme. • Presenters didn't use proper terminology.

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC

Rubrics for Project Proposal Assessment

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Timeliness (10%)	• All required elements are produced on time	• Most required elements are produced on time	• Few required elements are produced on time	• Required elements are not turned in
Presentation (20%)	• Proposal is well written with no grammatical or spelling errors • Proposal has been reviewed at least once • All required elements are included • Timelines and charts are written clearly, with no unnecessary marks or cross-outs	• Proposal is well written with few grammatical or spelling errors • Proposal has been reviewed at least once • Most required elements are included • Timelines and charts are written clearly with few unnecessary marks or cross outs	• Proposal is fairly well written with many grammatical or spelling errors • Proposal has never been reviewed • Most required elements are included • Timelines and charts are written clearly with many unnecessary marks and cross-outs	• Proposal is not well written and has many grammatical or spelling errors • Proposal has never been reviewed • Few required elements are included • Timelines and charts are not written clearly with many unnecessary marks and cross-outs
Practicality (30%)	• Proposed solution demonstrates understanding of real-world constraints (i.e. laws of physics, time) • Timeline specifies due dates for required deliverables • Materials list is reasonable, given resources • Proposal clearly links problem to proposed solution	• Proposed solution demonstrates a fair understanding of real-world constraints (i.e. laws of physics, time) • Timeline specifies most due dates for required deliverables • Materials list is mostly reasonable, given resources • The proposal mostly links problem to the proposed solution	• Proposed solution demonstrates a poor understanding of real-world constraints (i.e. laws of physics, time) • Timeline specifies few due dates for required deliverables • Materials list is not reasonable, given resources • Very little connection made between the proposed solution and the problem	• No proposed solution is given • No timeline indicated • Materials list is not reasonable • No connection made between the proposed solution and the problem

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC

Rubrics for Robotics Explorations Assessment

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Build/ Program Test Robot	• Robot is built accurately with no mistakes • Robot is programmed accurately with no errors	• Robot is built with few mistakes • Robot is programmed with few errors	• Robot is built with some mistakes • Robot is programmed with some errors	• Robot is built with many mistakes • Robot is programmed with many errors • Robot is built with many mistakes • Robot is programmed with many errors
Data Analysis/ Scientific Method	• Make reasonable predictions based on prior knowledge • Correctly gather and record all data • Accurately construct a bar graph illustrating results	• Make predictions based on prior knowledge • Correctly gather and record most data • Construct a bar graph illustrating results with few errors	• Predictions are irrational and not based on prior knowledge • Correctly gather and record some data • Construct a bar graph illustrating results with some errors	• No predictions made • Data is not gathered or recorded correctly • Bar graph is inaccurate
Writing an Analytical Paragraph	Paragraph includes: • A topic statement accurately presenting information on probability. • A detailed explanation of how the results of the trial compare with probability calculations. • A concluding statement thoroughly summarizing results of the experiment	Paragraph includes: • A topic statement accurately presenting information on probability. • An explanation of how the results of the trial compare with probability calculations. • A concluding statement summarizing results of the experiment.	Paragraph includes: • A topic statement presenting some information on probability. • Some explanation of how the results of the trial compare with probability calculations. • A concluding statement summarizing some results of the experiment.	Paragraph includes: • A topic statement that does not present information on probability. • Little explanation of how the results of the trial compare with probability calculations. • A concluding statement that does not summarize results of the experiment.

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC

Rubrics for Work Habits Evaluation

Student Name _____ Date _____

Course Title _____ Grading Period _____

10 Advanced 9 Proficient 8 Basic 7-0 Unacceptable

		Self	Teacher
1	Gives full attention to instructions and follows directions.		
2	Comes prepared and works the entire class period.		
3	Works well with minimal supervision.		
4	Works up to potential, shows maximum effort.		
5	Works cooperatively as a member of a group.		
6	Makes effective use of time and/or materials.		
7	Demonstrates initiative and motivation.		
8	Has a cooperative, positive attitude.		
9	Is on time for class.		
10	Participates daily in the cleanup program.		
	Work Habits Point Total		

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC

Rubrics for Workplace Competencies Evaluation

Student Name _____ Date _____

Course Title _____ Grading Period _____

Calculate a score based on the following workplace competencies

1	Consistently demonstrates a positive attitude Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
2	Cooperates all of the time Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
3	Communicates well Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
4	Honest Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
5	Dependable Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
6	Recognizes problems and finds acceptable solutions Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
7	Concentrates Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
8	Produces quality work Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
9	Makes smart decisions Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
10	Always attends Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
11	Always punctual Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
12	Follows directions Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
13	Reads, writes, and calculates well Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
14	Shows leadership Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
15	Practice good grooming and dresses appropriately Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
16	Meets deadlines while producing quality Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)	
	Work Habits Point Total	

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC

Rubrics for Robotics Class Writing

All students in robotics class will be using the writing process when completing assignments for class. Brainstorm your topic, outline what you plan to write about, then write your essay.

For full credit you will need to turn in proof of the following:

1. Brainstorm worksheet: A sheet of paper that includes all ideas generated during the brainstorming process. Include everything that you think of about your topic when brainstorming.
2. An outline of how you you plan to present your topic. Outline the information that you have in your brainstorming sheet. What will you talk about first, second, third, how will you support what you are writing about? Put it on the back of your brainstorming sheet.
3. 1st draft of your paper: Double-space your paper so that it will be easier to edit. You can edit on the same copy that you turn in. Just make sure that it can be read.

You will be evaluated on the following:

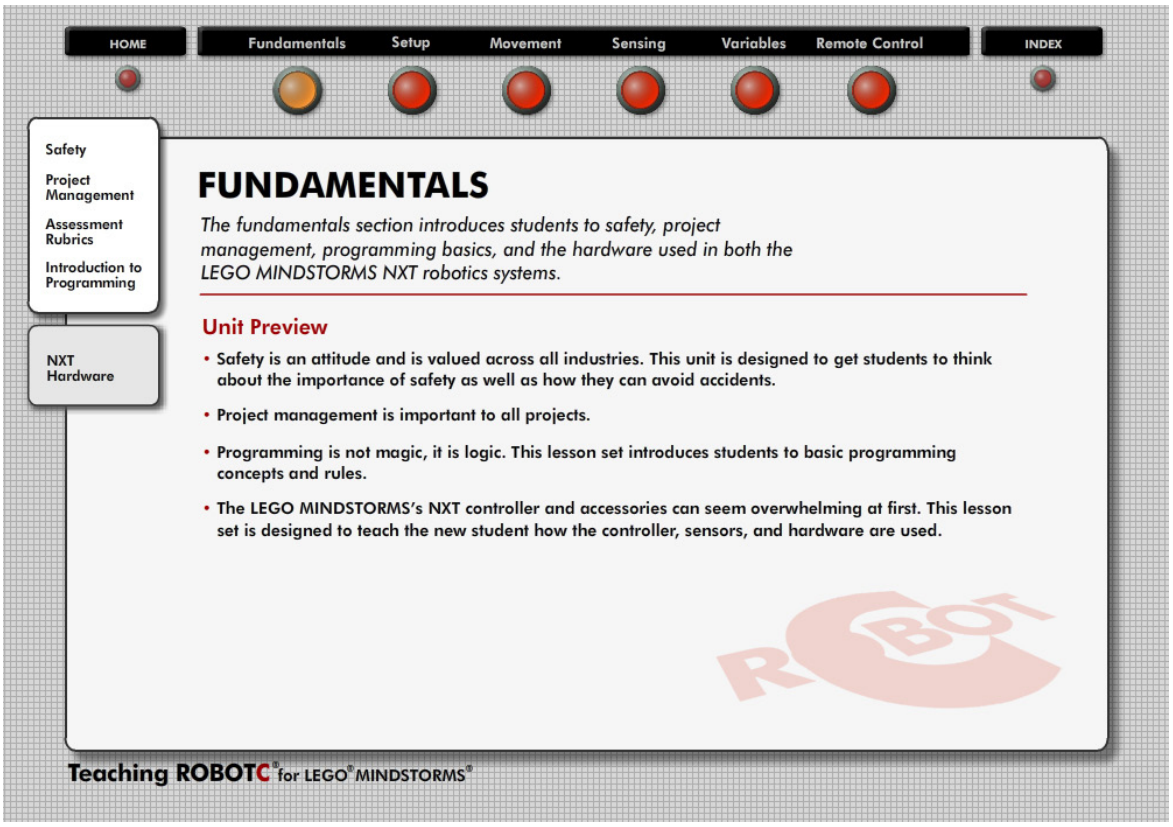
- One point for your brainstorm worksheet.
- One point for your outline.
- One point for the accuracy of the information being presented.
- One point for how well your paper reads. Is it choppy? Does the information you wrote clearly support what you are trying to say? Did you proofread your paper for errors?
- One point for spelling.

5pts A 4pts B 3pts C 2pts D

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

© 2009 Carnegie Mellon Robotics Academy

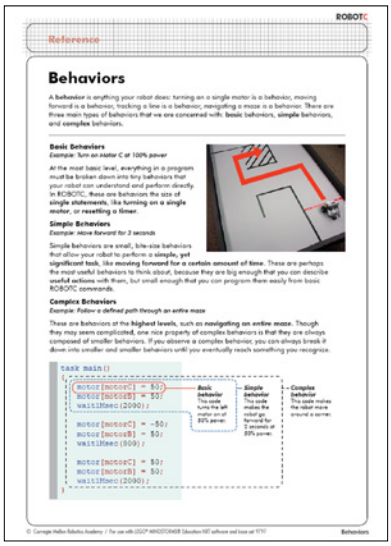
Getting Started/Fundamentals/Introduction to Programming

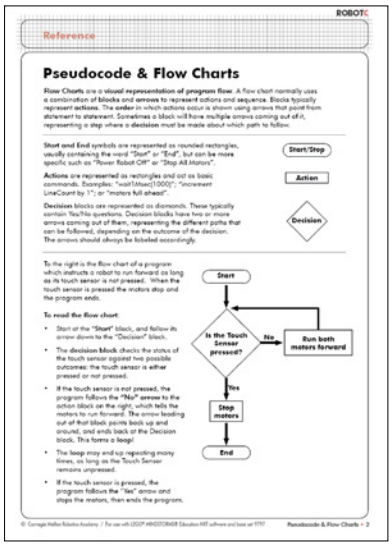


Introduction to Programming

Thinking About Programming Video Set - The "Programmer and Machine" and "Planning and Behaviors" videos explain to students the role of the programmer and the machine, and how the programmer must learn to think like a machine in order to program robots. The videos are designed to explain programming concepts to beginners.

Behaviors and Flowcharts & Pseudocode Helper Pages - These are reference PDFs that students can use as study guides. The helper pages are designed to be guides to the topics.





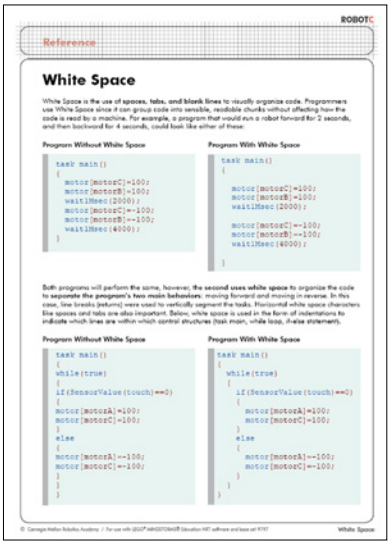
© 2009 Carnegie Mellon Robotics Academy

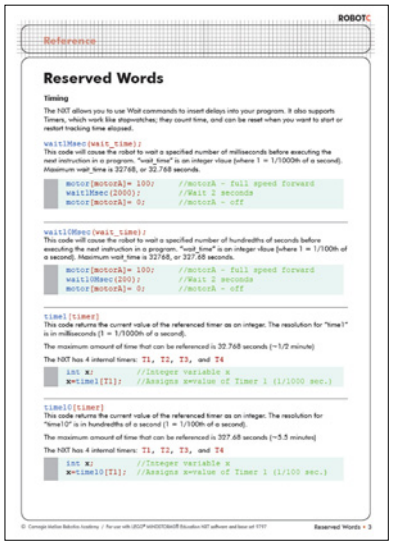
Getting Started/Fundamentals/Introduction to Programming

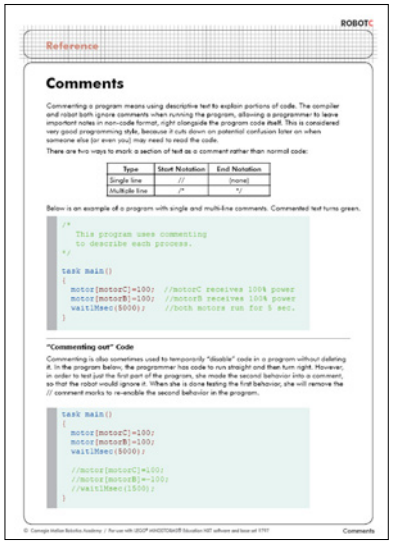
Thinking about Programming ROBOTC Rules Parts 1 & 2 - This video set builds the foundational knowledge that student must develop to begin to program using any syntax-based programming language. This video set introduces syntax concepts in a sequential and logical manner.

Whitespace, Comments, and Reserved Words Helper Pages - These are reference PDFs that students can use for reference or as study guides.

Thinking About Programming and ROBOTC Programming Quiz - One page quiz designed to check students understanding of introductory programming topics.

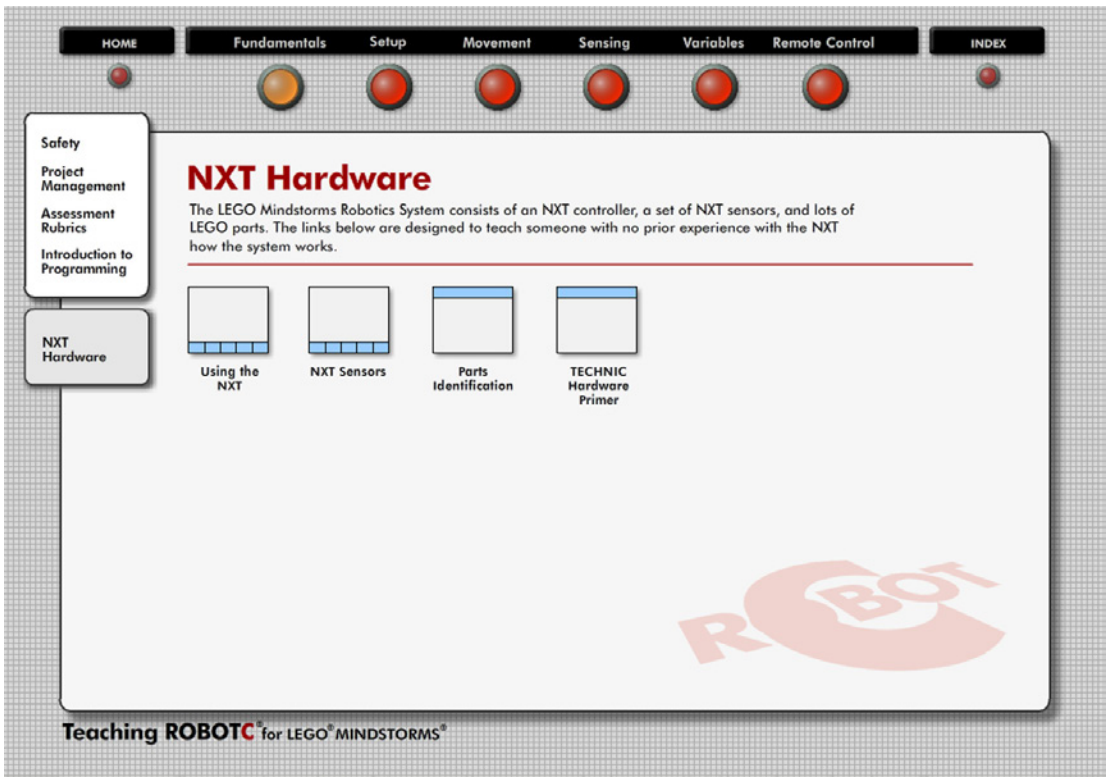






© 2009 Carnegie Mellon Robotics Academy

Getting Started/Fundamentals/NXT Hardware

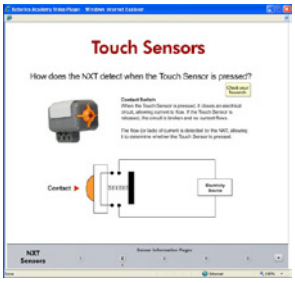


NXT Hardware

NXT Sensors Slide Show - The Sensors Slide Shows introduce new learners to the MINDSTORM sensors that are part of the base set: the smart motors, touch, light, ultrasonic and sound sensors. Students will learn how they work and what type of feedback they record.

NXT Parts Identification - Graphically introduces students to the parts and how they work.

TECHNIC Hardware Primer- The TECHNIC hardware primer is designed to show students how LEGO TECHNIC parts can be used to build subassemblies that they can use in their robot designs.

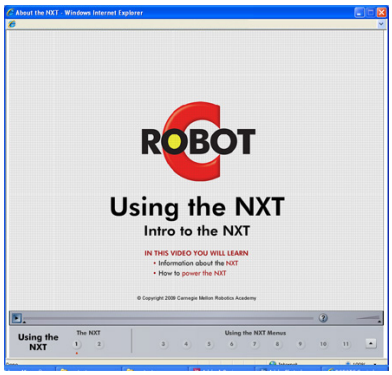


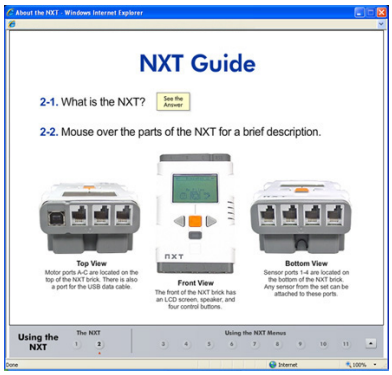
© 2009 Carnegie Mellon Robotics Academy

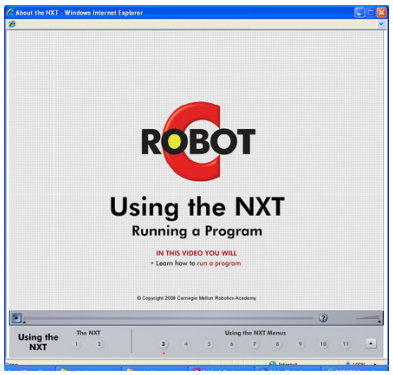
Getting Started/Fundamentals/NXT Hardware

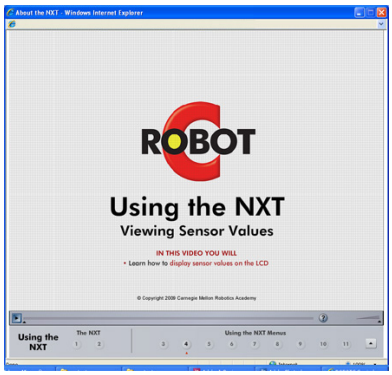
NXT Hardware

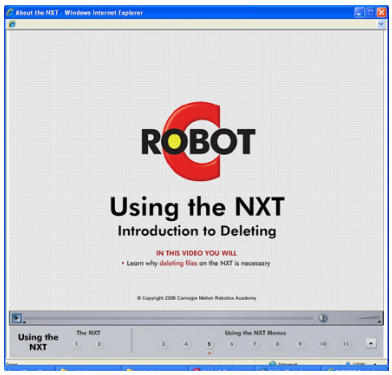
Using the NXT Slide Show - Assumes that the user has never used an NXT. There are 11 sections of the slide show that include: Introduction to the NXT, NXT Guide, Running a Program, Viewing Sensor Values, Introduction to Deleting, Deleting Programs 1, 2 & 3, Returning to the Main Menu, Changing the Volume, and Turning Off the NXT. This resource is a very valuable teaching tool for students just beginning to use the NXT.

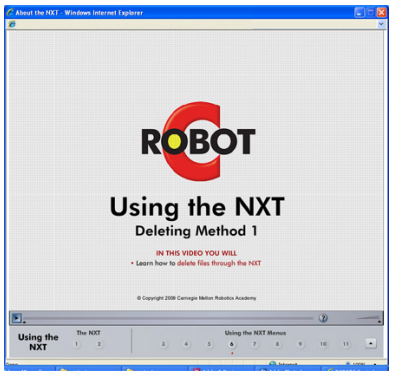


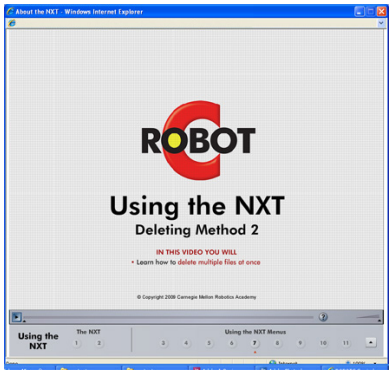


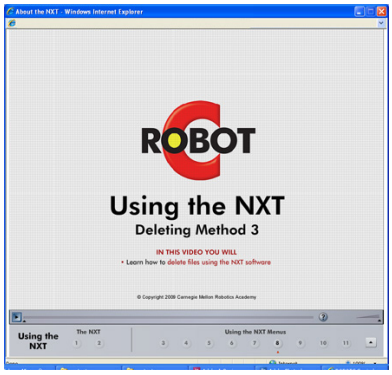


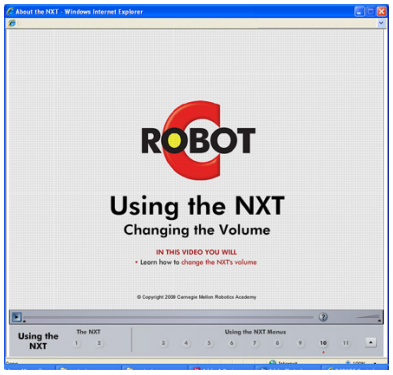


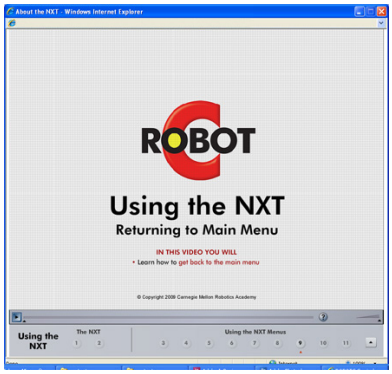


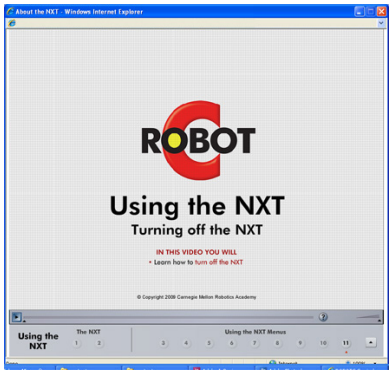






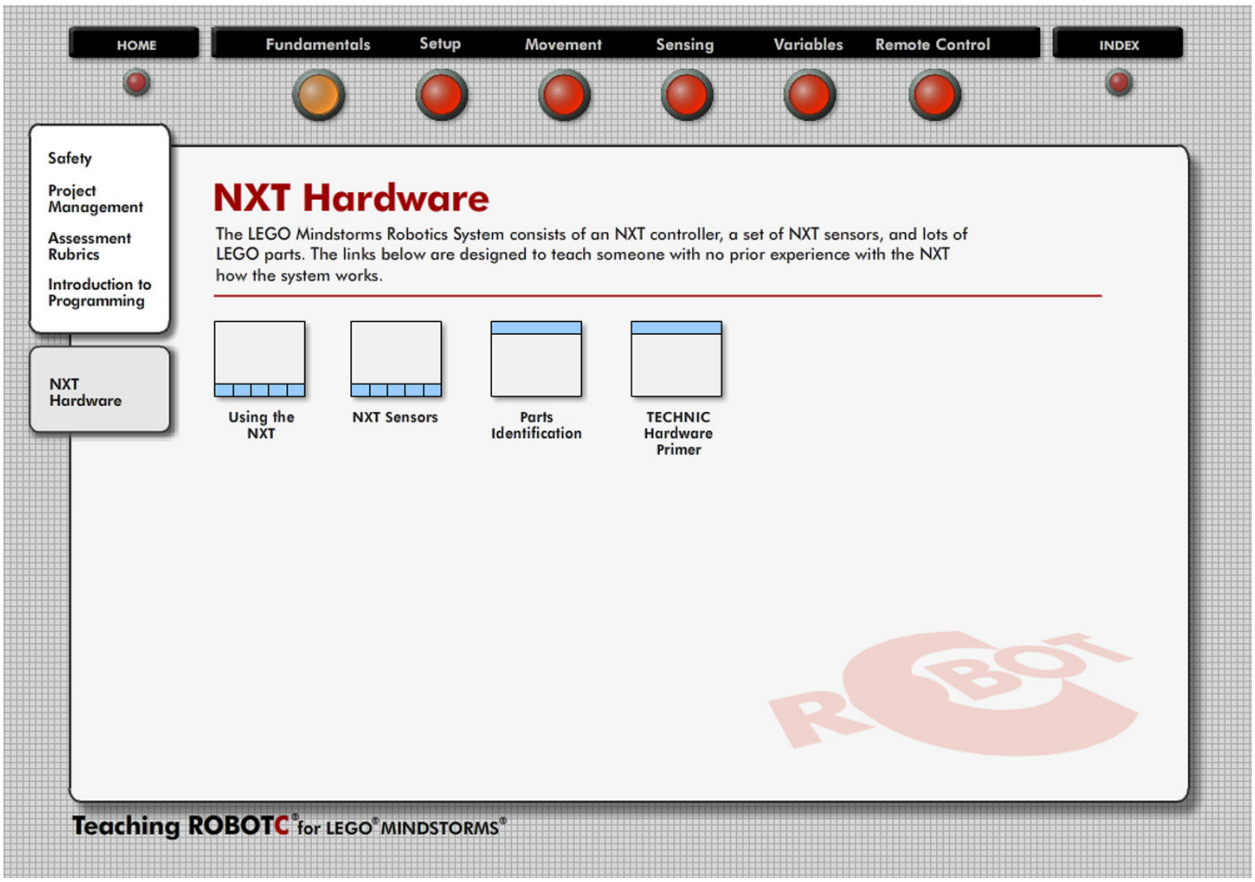






© 2009 Carnegie Mellon Robotics Academy

Getting Started/Setup/NXT Setup



NXT Setup Page

Build the REM Robot - All of the ROBOTC for MINDSTORMS lessons use the “Robot Educator Model” plus attachments. This page provides building instructions in both a printable and digital format.

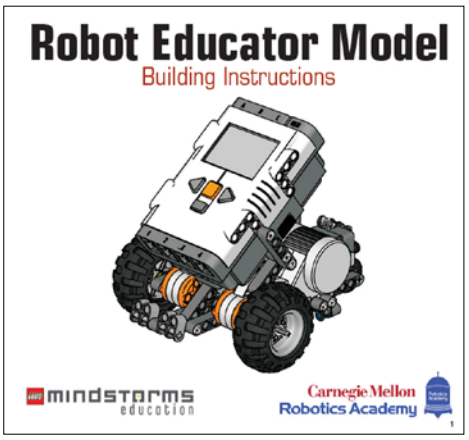
Download Firmware Video and Quiz - Students will learn what firmware is and how to download it to their robots. There is also a quiz designed to check student understanding.

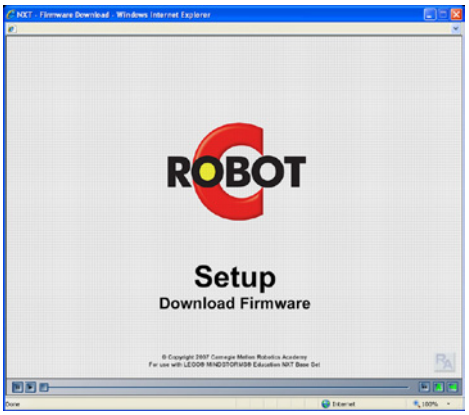
Download Sample Program and Quiz- Students will learn to download their first ROBOTC program and test it on their NXT. The lesson also includes a PDF helper page as well as a quiz designed to check student understanding.

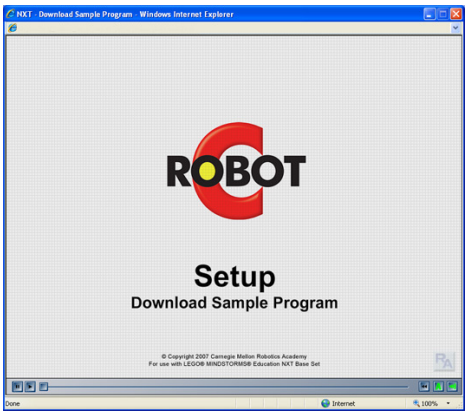
© 2009 Carnegie Mellon Robotics Academy

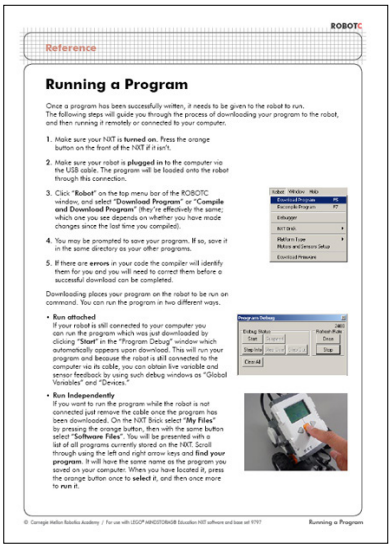
Getting Started/Setup/NXT Setup continued

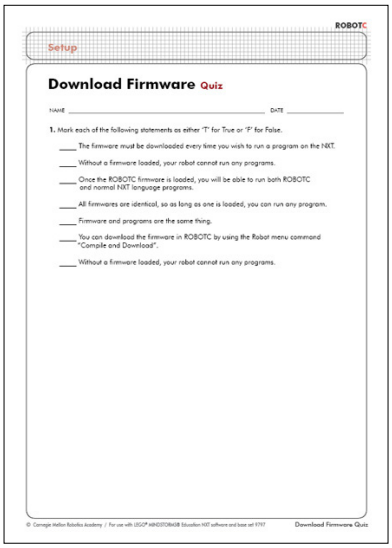
NXT Setup Page Resources Continued

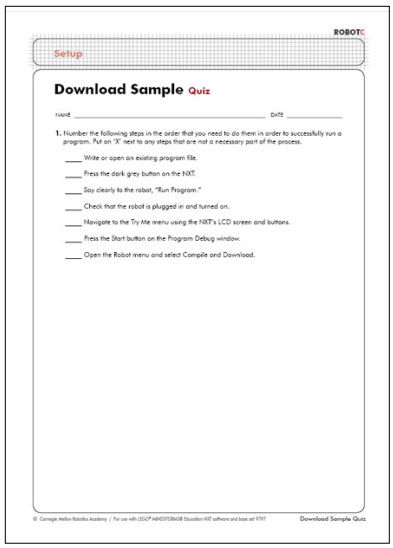






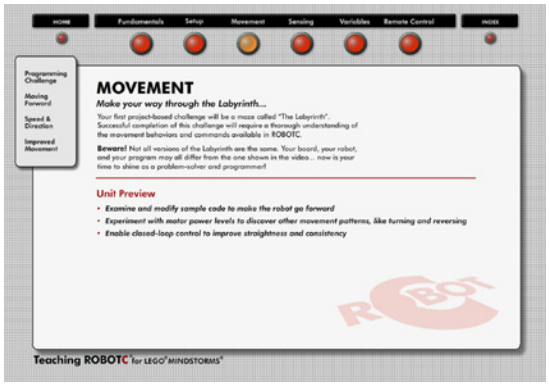


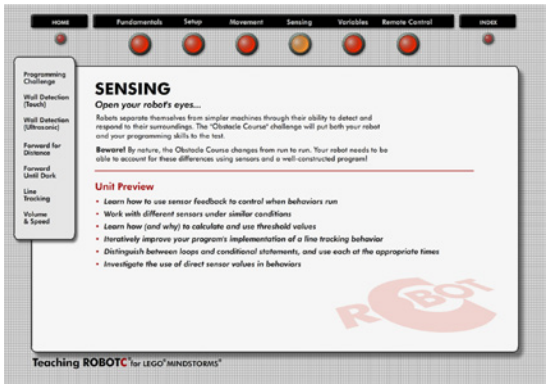


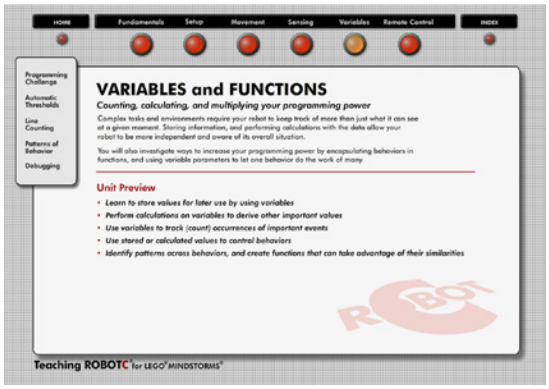


© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming…
Introduction to Engineering





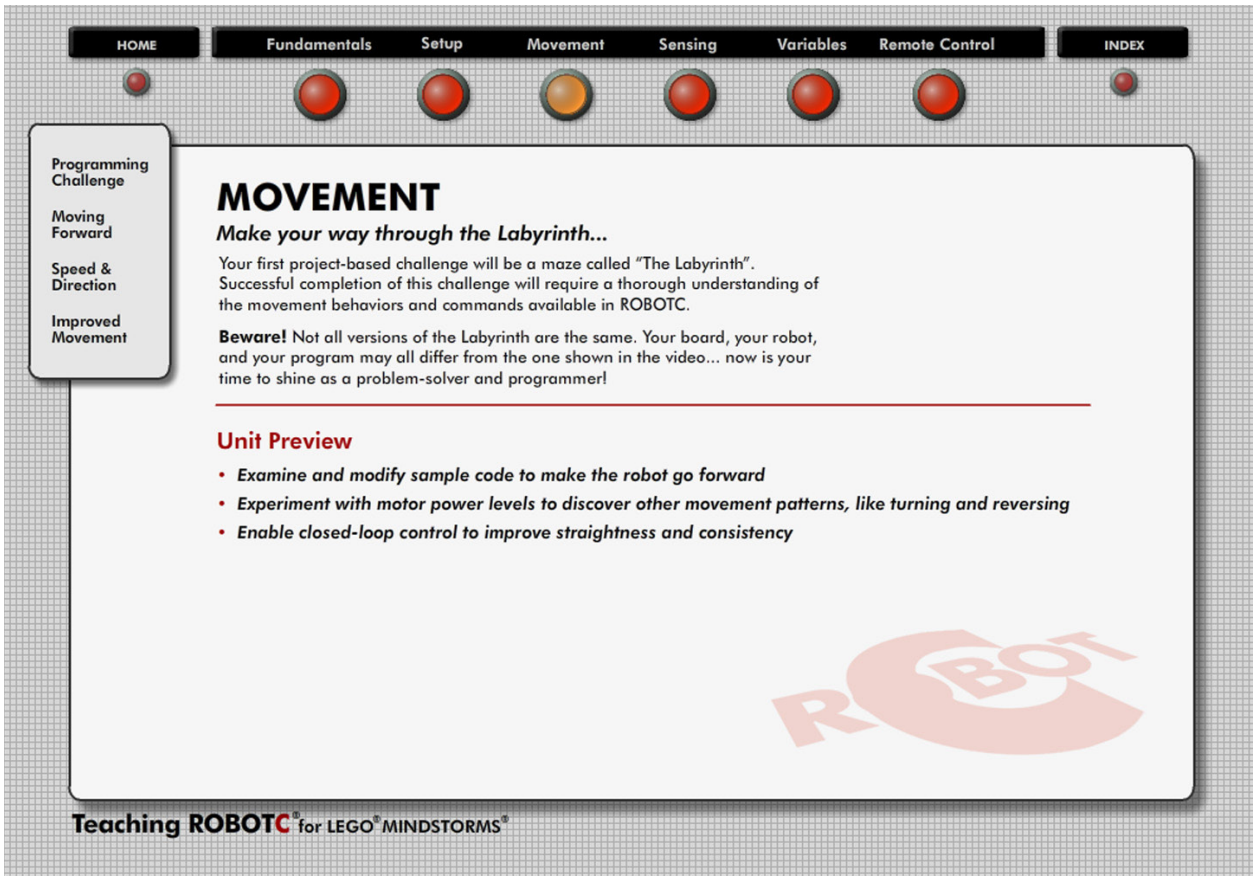


The Introduction to Programming Curriculum is divided into four instructional units: Movement, Sensing, Variables and Functions, and Remote Control. Each unit contains a number of Lesson Sets designed to teach a particular concept. Each unit is organized around a “Unit Programming Challenge”. The solution for the Unit Programming Challenge is taught as students move through the individual Lesson Sets.

The student will find other exercises designed to reinforce their programming skills or designed to give them a deeper understanding of the NXT hardware.

© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Movement/Programming Challenge



The Movement Unit

The Movement Unit is taught using three Lesson Sets and a programming challenge. The Lesson Sets begin using sample code that is already included in ROBOTC. The first Lesson Set, Moving Forward, teaches students in a very lockstep manner what each line of code does while introducing them to moving motors for specific amounts of time. The second Lesson Set, Speed and Direction, explains motor power levels and how to reverse polarity. The second Lesson Set includes several "engineering labs" that the students will complete. The engineering labs place students in the role of engineer where they run their robots, measure results, iteratively test the results to determine reliability, and then extrapolate from their data set to predict new robot behaviors. The third Lesson Set, Improved Movement, begins to teach very important lessons about PID control, motor synchronization, and setting motor targets.

The Movement Unit also includes several programming challenges where students are challenged to solve simple movement programming challenges.

It will be important to remind students that although the initial work may seam easy, that the skills that they learn in the movement unit are foundational pieces that they must understand before they move to the sensing unit.

© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Movement/Programming Challenge

Labyrinth Programming Challenge

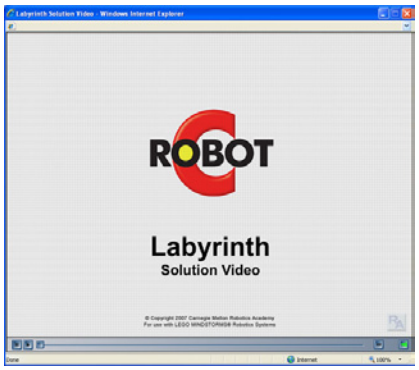
Each Programming Unit (Movement, Sensing, Variables and Remote Control) contains a programming challenge that is designed to place the learning into an interesting context. In the “Movement” Lesson Set the programming challenge is the “Labyrinth Challenge”.

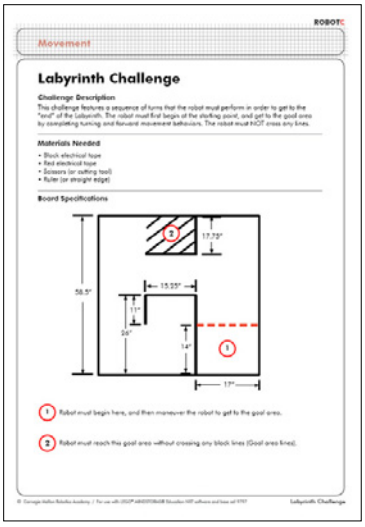
In this challenge, students will learn:

- Behavior based programming logic
- How to program their robots to accurately move forward, backward and turn
- The syntax rules related to programming using ROBOTC

Labyrinth Challenge Teacher Resources

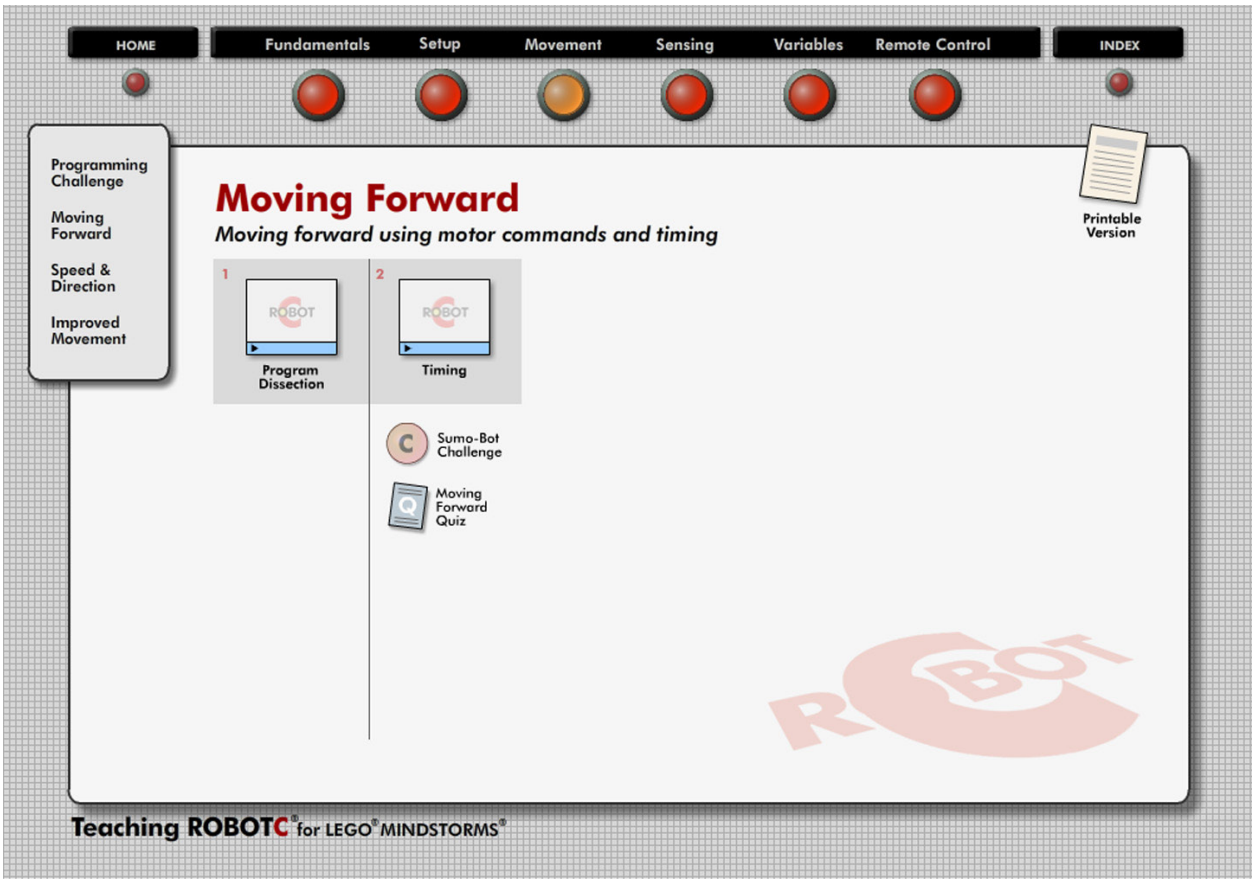
Each programming challenge comes with a PDF that explains the rules to the challenge as well as a video solution. These Labyrinth challenge resources are pictured on the left.





© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Movement/Moving Forward



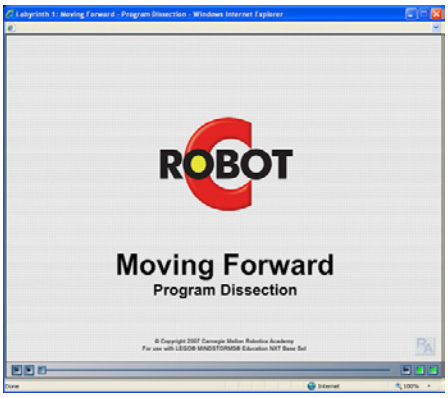
The Moving Forward Lesson Set

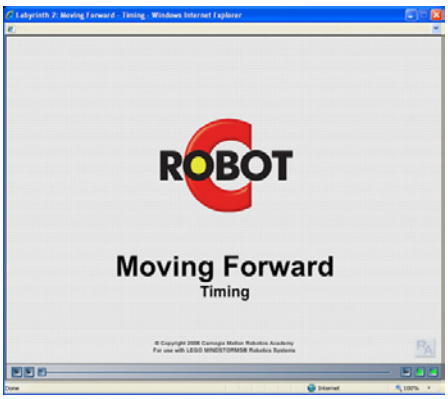
Each Lesson Set is designed to teach a related set of programming concepts, and is designed to give students confidence opening a program and understanding what the individual lines of code mean. Every set is supported with the following video and print resources:

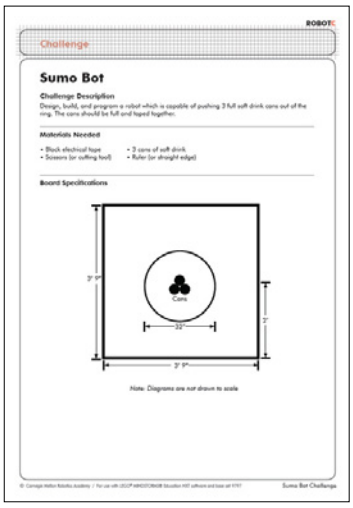
Program Dissection Video - Students are given a line by line description of the code used in the first sample program.

Timing Video - Timing offers the least accurate way of programming a robot to move from point to point. It is also the simplest method of programming. Students are introduced to wait-states.

Sumo-Bot Challenge - This fun challenge involves developing a programming/mechanical solution to push as many cans as possible out of a ring.

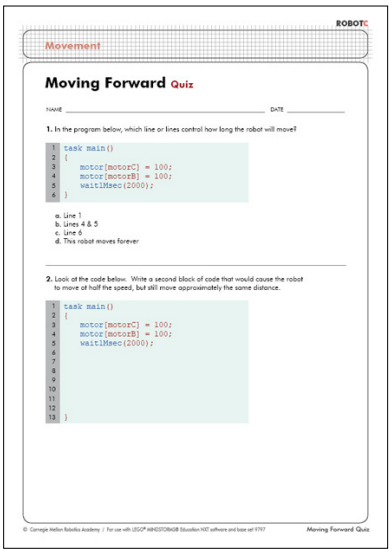


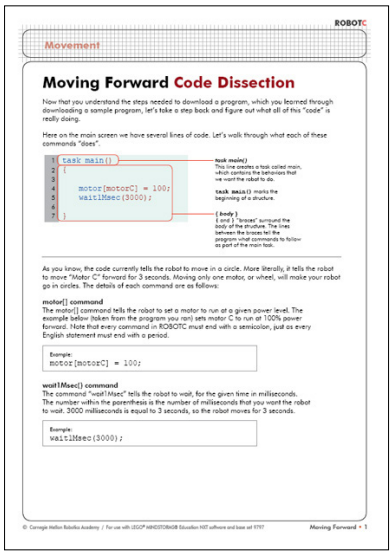




© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Movement/Moving Forward continued





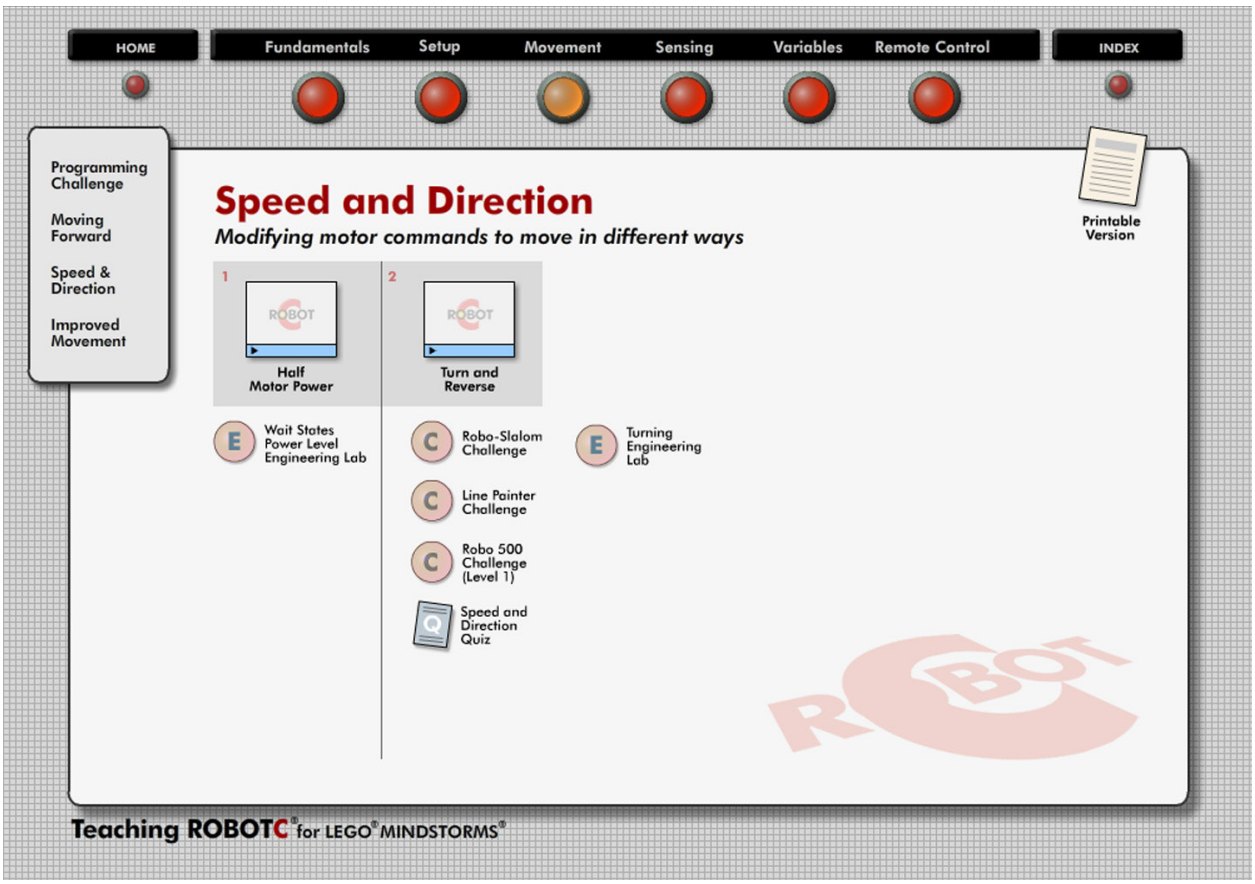
Additional Moving Forward Lesson Set Resources

Pictured at the left are two additional resources that accompany the Moving Forward Lesson Set.

The Moving Forward Quiz

The Moving Forward PDF - Each set of videos is accompanied by a PDF version of the script. The PDF version includes lots of pictures and text to describe everything that is shown in the videos. The PDF can be used either to accompany the video instruction or as a study guide.

Introduction to Programming/Movement/Speed and Direction



The Speed and Direction Lesson Set

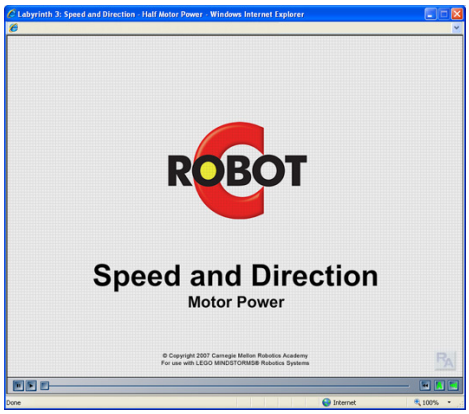
The Speed and Direction Lesson Set consists of two video lessons, two engineering challenges, three open-ended programming challenges, and a quiz. The resources are shown and explained in on the next page.

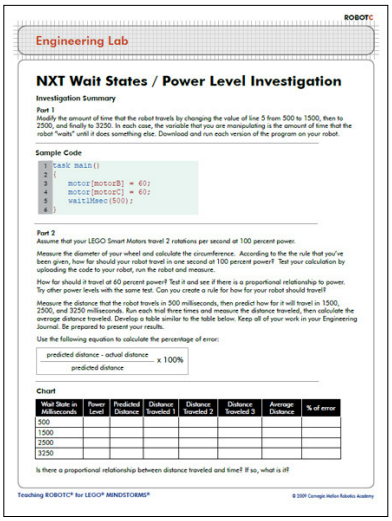
© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Movement/Speed and Direction cont.

Motor Power Lesson

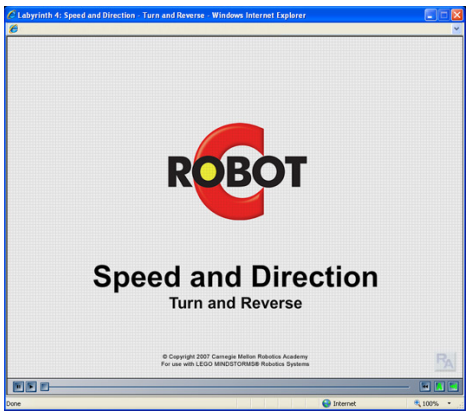
The first video lesson, Motor Power, teaches students how to change the power level on the robot. Students learn that as they change the power level, they are in effect changing the robot's speed. They will also complete an Engineering Lab named "NXT Wait States/Power Level Investigation". In this investigation, students program their robots at a variety of power levels and keep the amount of time the robot runs constant. In the investigation students identify if there is a proportional relationship between power levels and speed.

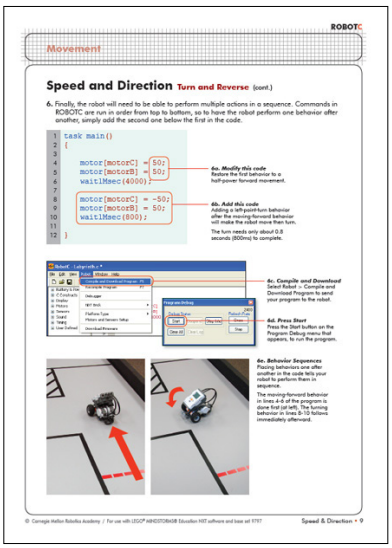


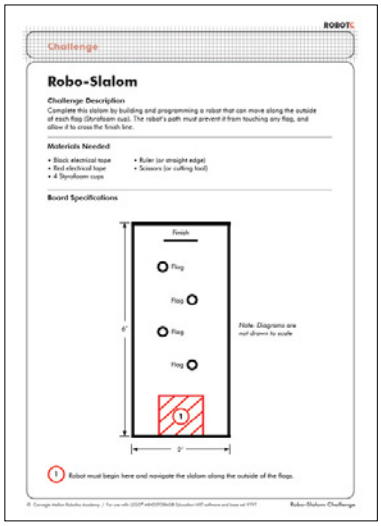


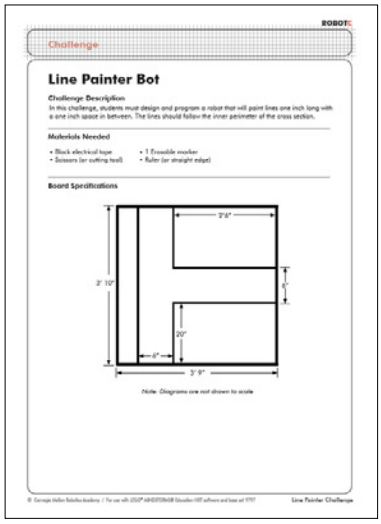
Turn and Reverse Lesson

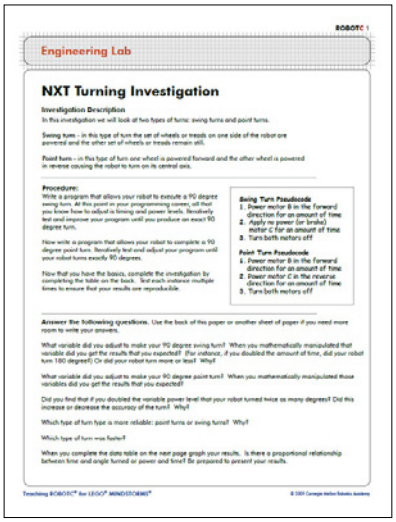
In the "Turn and Reverse" lesson, students learn about polarity. The Lesson Set includes a 10 page PDF, shown at the right, designed to complement the videos. It also includes 3 open ended programming challenges. Two of the programming challenges are shown below: Robo-Slolom and Line Painter Bot. Students complete the lesson with an Engineering Challenge, where they are asked to iteratively test robot turns given a specific power level and a target angle and then use that data to predict what the "wait state" will be to complete other turns.





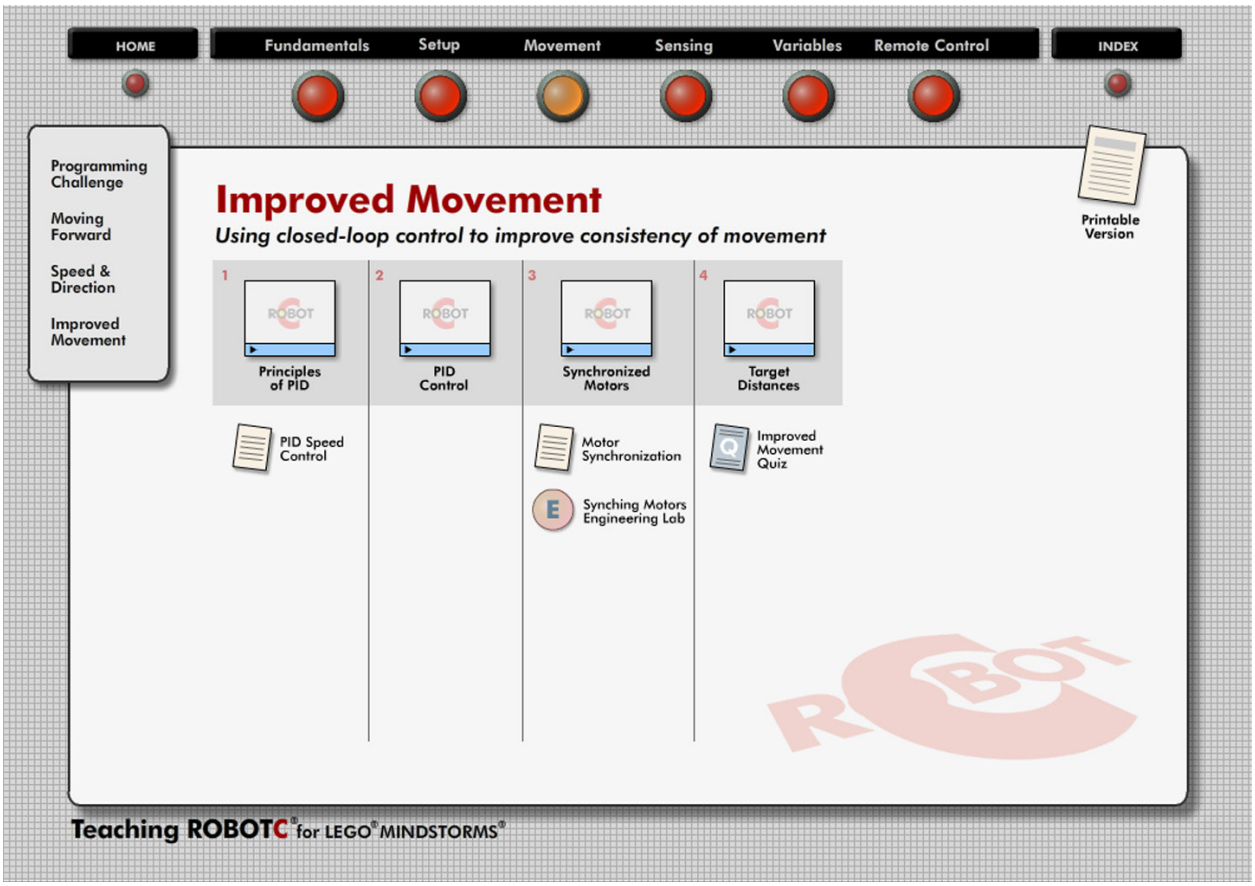






© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Movement/Improved Movement



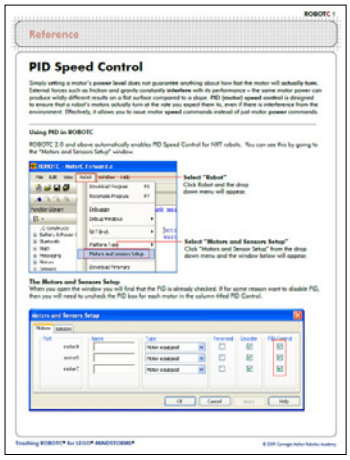
Improved Movement Lesson Set

The first half of the Improved Movement Lesson Set teaches students about PID speed control as well as how to us the Motor and Sensor Setup Window, the Debug Windows, and the NXT Devices window. The following resources are available to teach PID control:

The Principles of PID Video - This video teaches students what PID (Proportional Integral Derivatives) is and how it uses a closed loop control system to regulate speed control.

The PID Control Video - This video teaches students how to use ROBOTC's debugger to see the NXT's speed regulation. Students will use the debugger windows to watch the internal feedback mechanisms built into the system.

PID Speed Control Reference Page - This two page handout explains PID, and shows how to monitor PID using ROBOTC's debugging window.



© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Movement/Improved Movement cont.

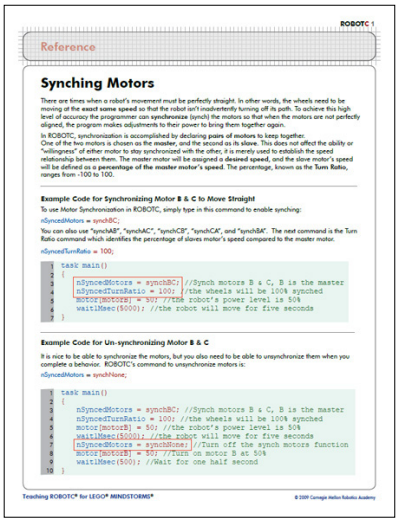
Synchronized Motors Lesson Set

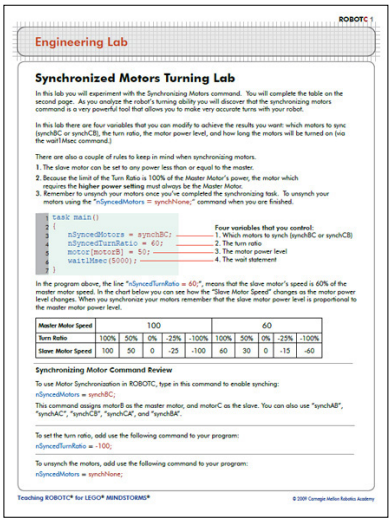
ROBOTC enables the programmer to synchronize the robot's wheel movements. The Lesson Set contains the following resources:

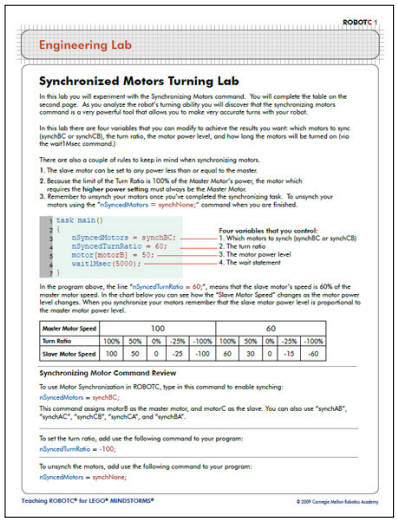
Synchronized Motors Video - Teaches students how to use ROBOTC reserved words to control their robots.

Synchronized Motor Reference Page - Can be used as a handout for a study guide or as a reference page.

Synchronized Motors Engineering Lab - Students perform a set of iterative experiments where they test robot performance and collect data designed to promote understanding of robot synchronization.





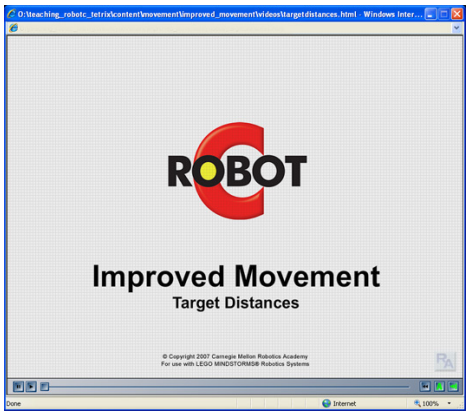


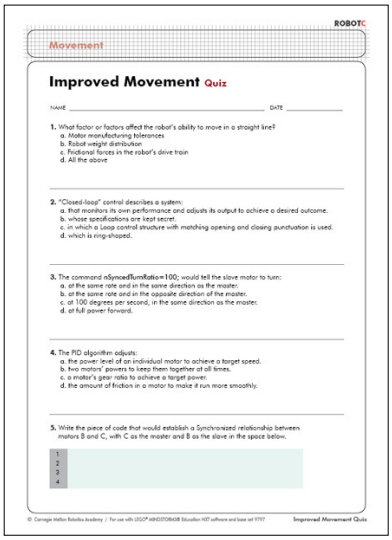
nMotorEncoder Command and Target Distances Lesson Set

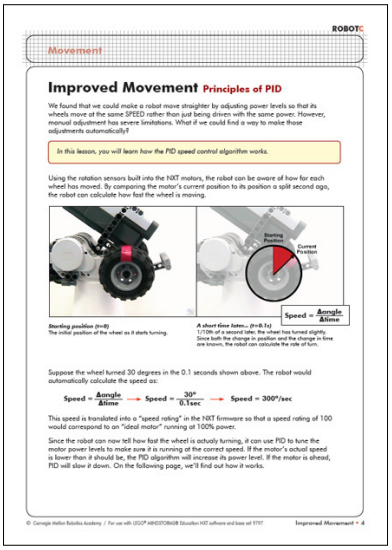
Encoders are used by the robot to count how many times a wheel on the robot turns. Students are introduced to encoders in this lesson allowing them to achieve better control of their robot.

The Target Distance Video - Teaches students how encoders work and gives them a naive understanding of the nMotorEncoderTarget function.

Improved Movement Lesson Set PDF - This PDF complements the videos used to teach the Improved Movement Lesson Set.

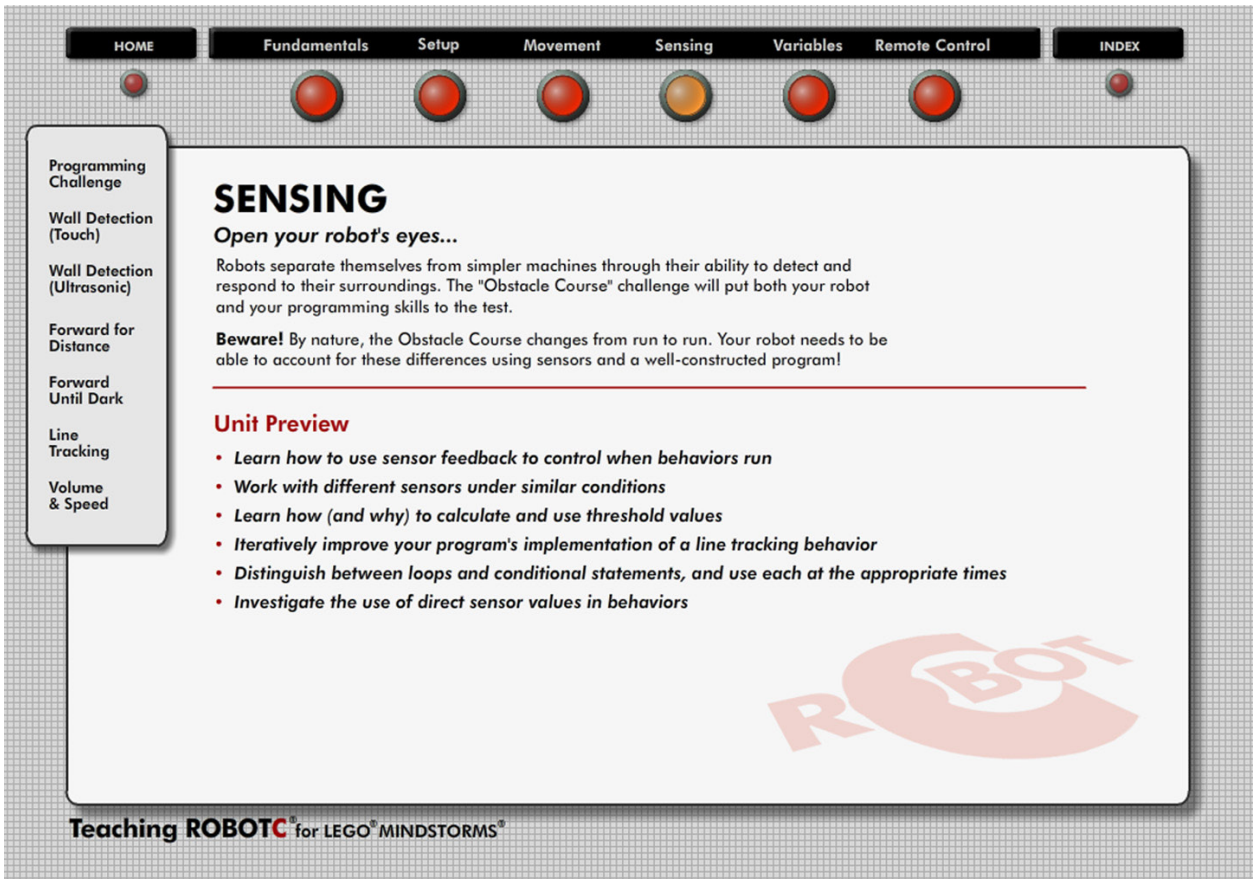






© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing



Sensing

The Sensing Unit consists of six Lesson Sets for the NXT. The lessons are designed to be taught in a sequential order from top to bottom. The lessons start with the touch sensor because of its simplicity, then move to the ultrasonic and sound sensors. If students skip one of the Lesson Sets, then they run the risk of missing a programming concept that was taught in a prior lesson. Do the lessons in sequence.

© 2009 Carnegie Mellon Robotics Academy

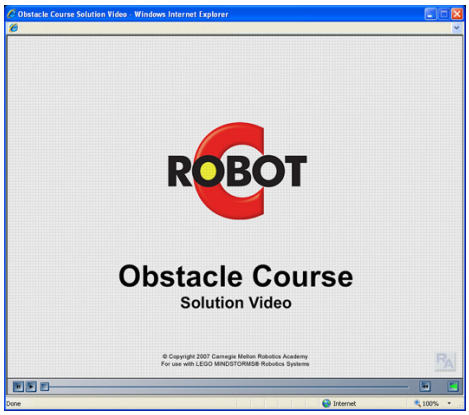
Introduction to Programming/Sensing/Programming Challenge

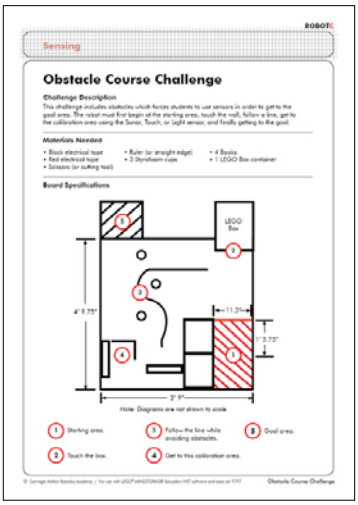
The Obstacle Course Programming Challenge

Students are required to use feedback from all sensors to complete the obstacle course. There is more than one way to solve this programming challenge, and individual students will develop their own programming style. For beginners, it will be important for the teacher to reinforce breaking robot behaviors down into their simplest parts.

The Obstacle Course Video - Shows a video solution to the programming challenge.

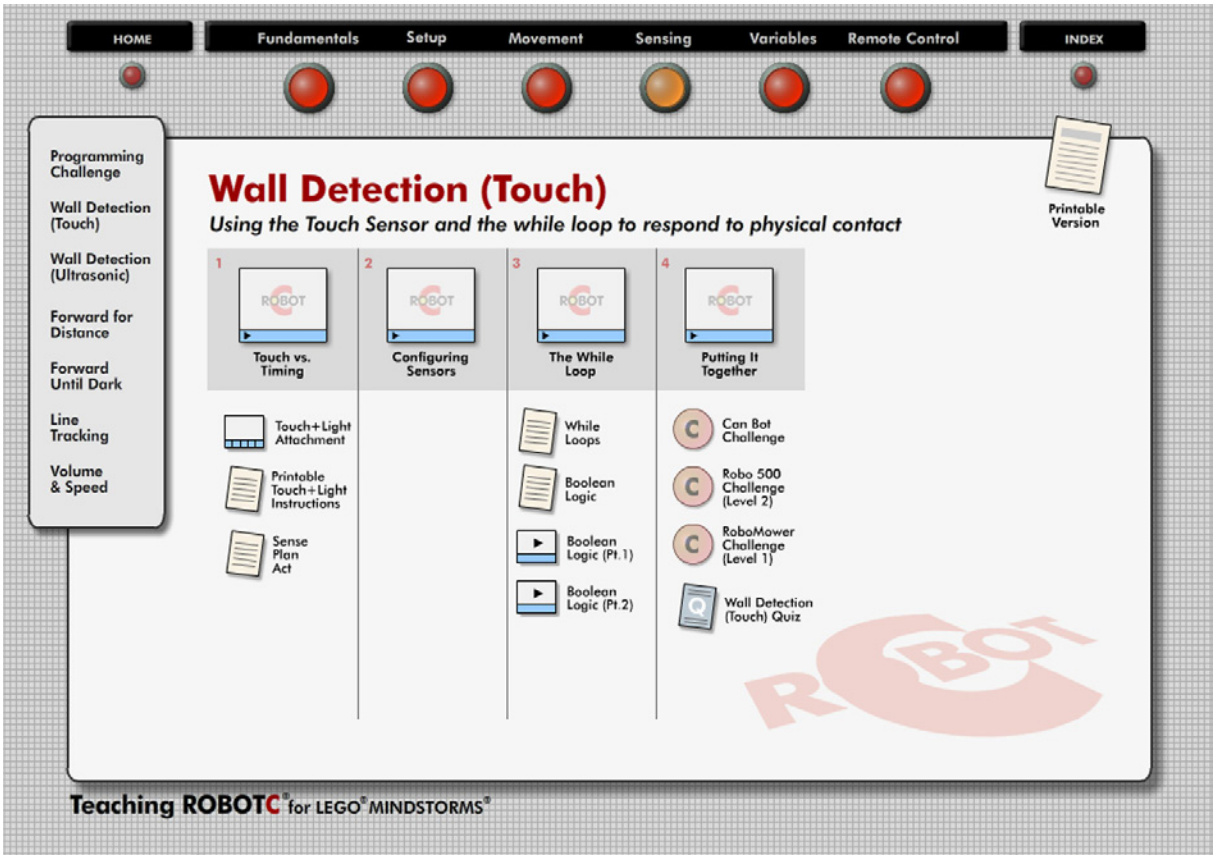
The Obstacle Course Handout - PDF that describes the challenge in terms of robot behaviors.





© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Touch Sensor



Sensing - Touch Sensor

The Wall Detection Lesson Set provides a comprehensive set of materials designed to introduce students to sensors. Students begin to solve the obstacle course by learning how a touch sensor works, how to configure the touch sensor using the motors and sensors setup windows, how to name sensors, and how to build a new program from scratch. As they learn about touch sensors, they will also learn about while loops, structures, boolean logic, conditional statements, ROBOTC reserved words, and how to use ROBOTC's built-in debugger windows to see sensor feedback values. Students will have access to the following video and print resources:

The Touch Versus Timing Video - This video introduces the touch sensor and explains how it will be used to help solve the obstacle course challenge.

The Configuring Sensors Video - This video shows students how to configure sensors using ROBOTC's Motors and Sensors Configuration Menu. It also shows students how to use the debugger screens to see sensor feedback.

The While Loop Video - Teaches students about conditional statements and how they control the while loop structure.

The Putting It All Together Video - This video is a step-by-step tutorial that begins with a blank program, then builds a program starting with configuring the sensors through setting up conditional statements and controlling motors.

Touch and Light Sensor Attachment - Instructional slide show that shows how to connect the sensor to the robot.

Sense Plan Act Reference Page - One page handout that describes the Sense-Plan-Act algorithm.

While Loop Reference Page - One page handout that describes the "while loop" structure.

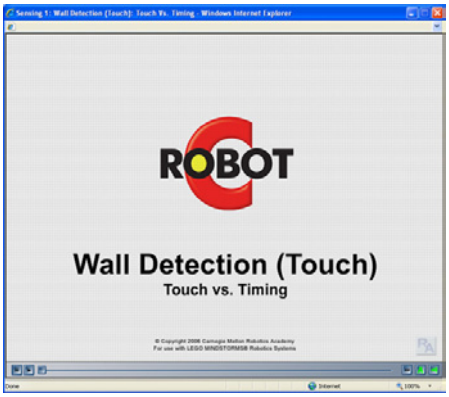
Boolean Logic Videos 1 & 2 - The first video teaches students how conditional statements work. The second teaches them about logical operators. There is also a Boolean Logic 3 page PDF that complements the videos.

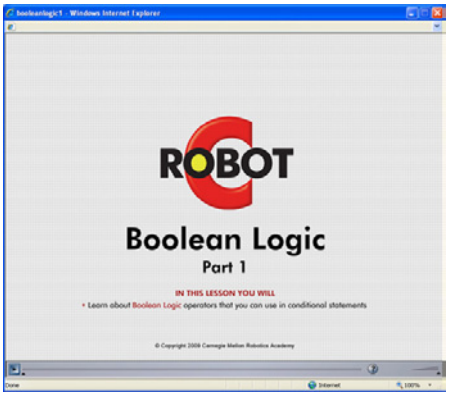
Robot Programming Challenges - This Lesson Set includes three open-ended programming challenges that can be solved by students: the Can Bot Challenge, Robo500 Challenge (Level 1) and the RoboMower Challenge (Level 1).

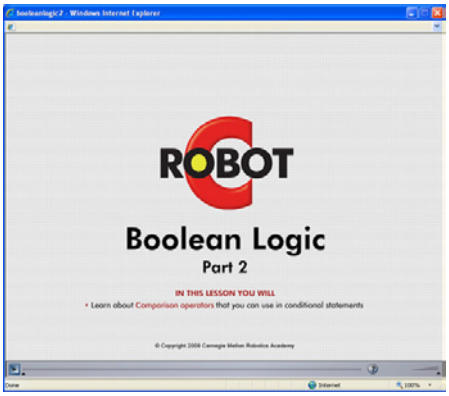
© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Touch Sensor

Sensing - Touch Sensor - Video Resources



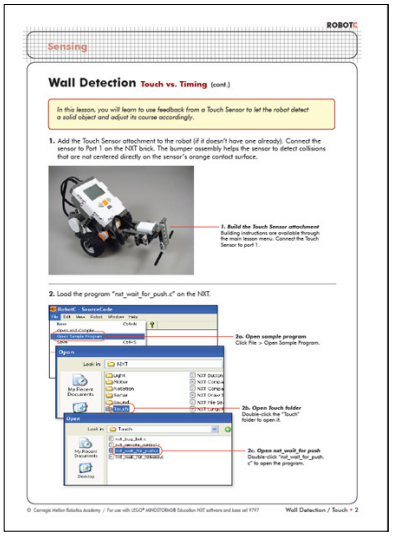




© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Touch Sensor

Sensing - Touch Sensor - Print Resources



ROBOTC

Sensing

Wall Detection Touch vs. Timing (cont.)

In this lesson, you will learn to use feedback from a Touch Sensor to let the robot detect a solid object and adjust its course accordingly.

1. Add the Touch Sensor attachment to the robot (if it doesn't have one already). Connect the sensor to Port 1 on the NXT brick. The bumper assembly helps the sensor to detect collisions that are not centered directly on the sensor's orange contact surface.

1. *Build the Touch Sensor attachment*
Building instructions are available through the main toolbar menu. Connect the Touch Sensor to port 1.

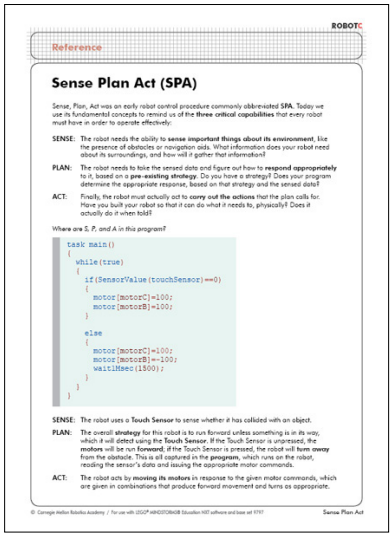
2. Load the program "*nxt_wait_for_push.c*" on the NXT.

2a. *Open sample program*
Click File > Open Sample Program.

2b. *Open Touch folder*
Double-click the "Touch" folder to open it.

2c. *Open nxt_wait_for_push*
Double-click "nxt_wait_for_push.c" to open the program.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Wall Detection / Touch • 2



ROBOTC

Reference

Sense Plan Act (SPA)

Sense, Plan, Act was an early robot control procedure commonly abbreviated **SPA**. Today we use its fundamental concepts to remind us of the **three critical capabilities** that every robot must have in order to operate effectively:

SENSE: The robot needs the ability to **sense important things about its environment**, like the presence of obstacles or navigation aids. What information does your robot need about its surroundings, and how will it gather that information?

PLAN: The robot needs to take the sensed data and figure out how to **respond appropriately** to it, based on a **pre-existing strategy**. Do you have a strategy? Does your program determine the appropriate response, based on that strategy and the sensed data?

ACT: Finally, the robot must actually act to **carry out the actions** that the plan calls for. Have you built your robot so that it can do what it needs to, physically? Does it actually do it when told?

Where are S, P, and A in this program?

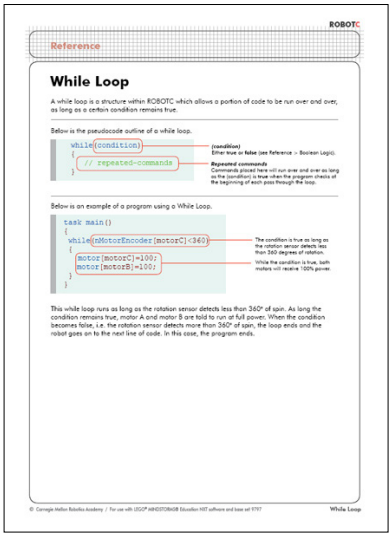
```
task main()
{
  while(true)
  {
    if(SensorValue(touchSensor)==0)
    {
      motor[motorC]=100;
      motor[motorB]=100;
    }
    else
    {
      motor[motorC]=100;
      motor[motorB]=-100;
      wait1Msec(1500);
    }
  }
}
```

SENSE: The robot uses a **Touch Sensor** to sense whether it has collided with an object.

PLAN: The overall **strategy** for this robot is to run forward unless something is in its way, which it will detect using the **Touch Sensor**. If the Touch Sensor is unpressed, the **motors** will be run **forward**; if the Touch Sensor is pressed, the robot will **turn away** from the obstacle. This is all captured in the **program**, which runs on the robot, reading the sensor's data and issuing the appropriate motor commands.

ACT: The robot acts by **moving its motors** in response to the given motor commands, which are given in combinations that produce forward movement and turns as appropriate.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Sense Plan Act



ROBOTC

Reference

While Loop

A while loop is a structure within ROBOTC which allows a portion of code to be run over and over, as long as a certain condition remains true.

Below is the pseudocode outline of a while loop.

```
while(condition)
{
  // repeated-commands
}
```

(condition)
Either **true** or **false** (see Reference > Boolean Logic).

Repeated commands
Commands placed here will run over and over as long as the (condition) is **true** when the program checks at the beginning of each pass through the loop.

Below is an example of a program using a While Loop.

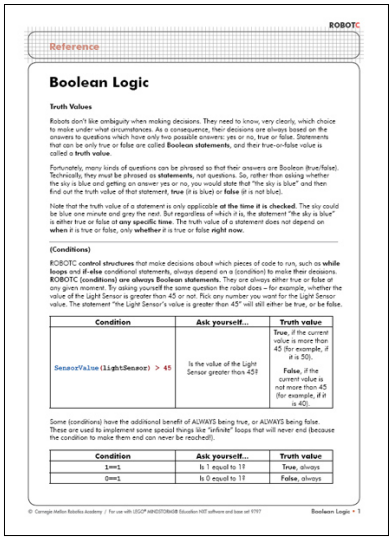
```
task main()
{
  while(nMotorEncoder[motorC]<360)
  {
    motor[motorC]=100;
    motor[motorB]=100;
  }
}
```

The condition is true as long as the rotation sensor detects less than 360 degrees of rotation.

While the condition is true, both motors will receive 100% power.

This while loop runs as long as the rotation sensor detects less than 360° of spin. As long the condition remains true, motor A and motor B are told to run at full power. When the condition becomes false, i.e. the rotation sensor detects more than 360° of spin, the loop ends and the robot goes on to the next line of code. In this case, the program ends.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 While Loop



ROBOTC

Reference

Boolean Logic

Truth Values

Robots don't like ambiguity when making decisions. They need to know, very clearly, which choice to make under what circumstances. As a consequence, their decisions are always based on the answers to questions which have only two possible answers: yes or no, true or false. Statements that can be only true or false are called **Boolean statements**, and their true-or-false value is called a **truth value**.

Fortunately, many kinds of questions can be phrased so that their answers are Boolean (true/false). Technically, they must be phrased as **statements**, not questions. So, rather than asking whether the sky is blue and getting an answer yes or no, you would state that "The sky is blue" and then find out the truth value of that statement, **true** (it is blue) or **false** (it is not blue).

Note that the truth value of a statement is only applicable **at the time it is checked**. The sky could be blue one minute and grey the next. But regardless of which it is, the statement "the sky is blue" is either true or false at **any specific time**. The truth value of a statement does not depend on **when** it is true or false, only **whether** it is true or false **right now**.

(Conditions)

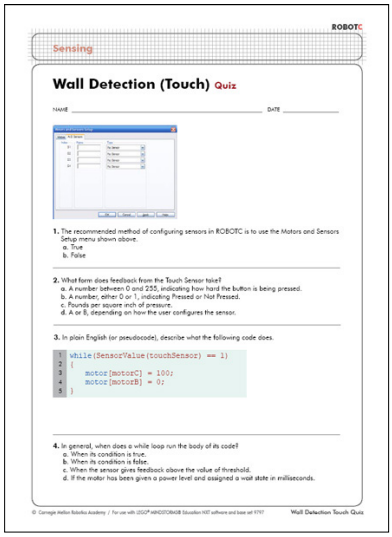
ROBOTC **control structures** that make decisions about which pieces of code to run, such as **while loops** and **if-else** conditional statements, always depend on a (condition) to make their decisions. ROBOTC **(conditions) are always Boolean statements**. They are always either true or false at any given moment. Try asking yourself the same question the robot does – for example, whether the value of the Light Sensor is greater than 45 or not. Pick any number you want for the Light Sensor value. The statement "the Light Sensor's value is greater than 45" will still either be true, or be false.

Condition	Ask yourself...	Truth value
`SensorValue(LightSensor) > 45`	Is the value of the Light Sensor greater than 45?	**True**, if the current value is more than 45 (for example, if it is 50). **False**, if the current value is not more than 45 (for example, if it is 40).

Some (conditions) have the additional benefit of ALWAYS being true, or ALWAYS being false. These are used to implement some special things like "infinite" loops that will never end (because the condition to make them end can never be reached).

Condition	Ask yourself...	Truth value
1==1	Is 1 equal to 1?	**True**, always
0==1	Is 0 equal to 1?	**False**, always

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Boolean Logic • 1



ROBOTC

Sensing

Wall Detection (Touch) Quiz

NAME _____ DATE _____

1. The recommended method of configuring sensors in ROBOTC is to use the Motors and Sensors Setup menu shown above.
   a. True
   b. False

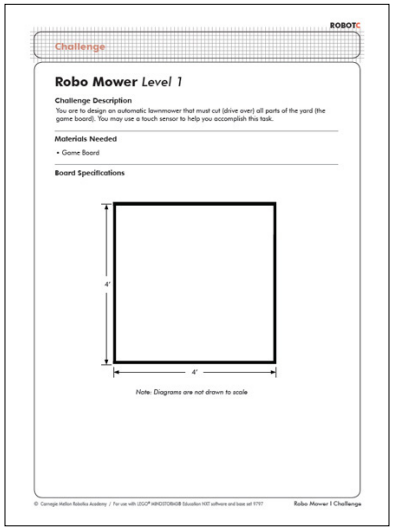
2. What form does feedback from the Touch Sensor take?
   a. A number between 0 and 255, indicating how hard the button is being pressed.
   b. A number, either 0 or 1, indicating Pressed or Not Pressed.
   c. Pounds per square inch of pressure.
   d. A or B, depending on how the user configures the sensor.

3. In plain English (or pseudocode), describe what the following code does.

```
while(SensorValue(touchSensor) == 1)
{
  motor[motorC] = 100;
  motor[motorB] = 0;
}
```

4. In general, when does a while loop run the body of its code?
   a. When its condition is true.
   b. When its condition is false.
   c. When the sensor gives feedback above the value of threshold.
   d. If the motor has been given a power level and assigned a wait state in milliseconds.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Wall Detection Touch Quiz



ROBOTC

Challenge

Robo Mower *Level 1*

Challenge Description

You are to design an automatic lawnmower that must cut (drive over) all parts of the yard (the game board). You may use a touch sensor to help you accomplish this task.

Materials Needed

- Game Board

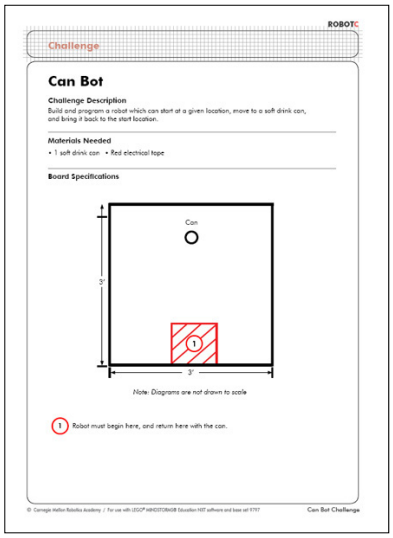
Board Specifications

4'

4'

Note: Diagrams are not drawn to scale

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Robo Mower I Challenge



ROBOTC

Challenge

Can Bot

Challenge Description

Build and program a robot which can start at a given location, move to a soft drink can, and bring it back to the start location.

Materials Needed

- 1 soft drink can
- Red electrical tape

Board Specifications

Can

3'

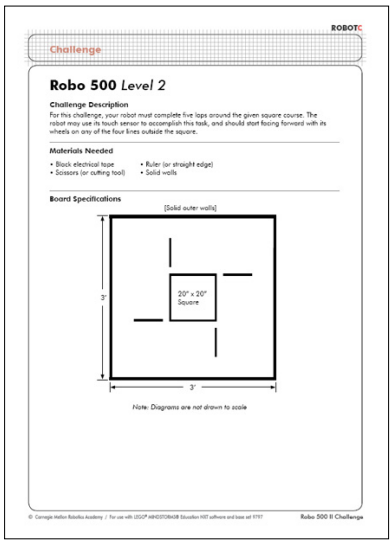
3'

1

Note: Diagrams are not drawn to scale

1 Robot must begin here, and return here with the can.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Can Bot Challenge



ROBOTC

Challenge

Robo 500 *Level 2*

Challenge Description

For this challenge, your robot must complete five laps around the given square course. The robot may use its touch sensor to accomplish this task, and should start facing forward with its wheels on any of the four lines outside the square.

Materials Needed

- Black electrical tape
- Scissors (or cutting tool)
- Ruler (or straight edge)
- Solid walls

Board Specifications

[Solid outer walls]

20" x 20" Square

3'

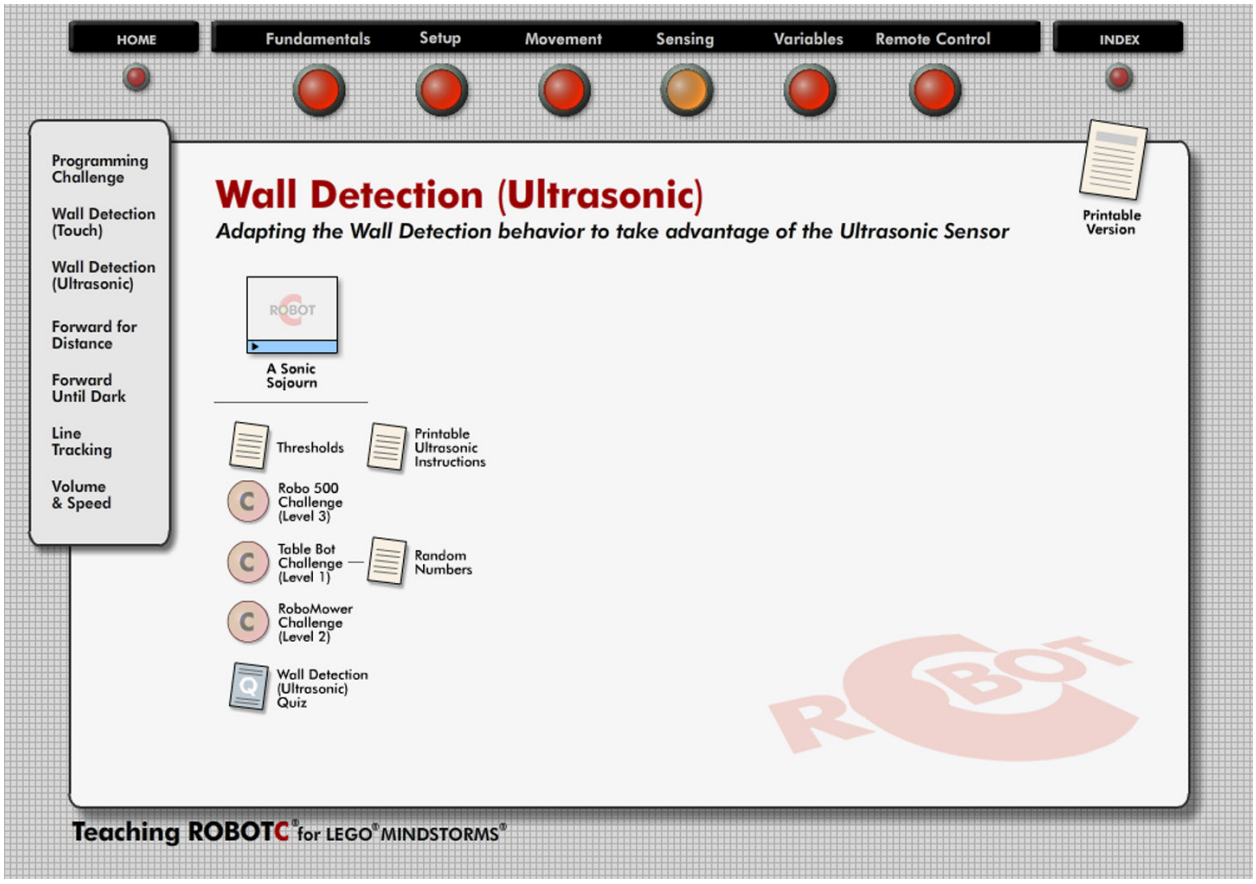
3'

Note: Diagrams are not drawn to scale

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Robo 500 II Challenge

© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Ultrasonic Sensor



Sensing - Ultrasonic Sensor

The Wall Detection Ultrasonic Sensor Lesson Set builds on what students learned in the Wall Detection Touch Sensor Lesson Set. This Lesson Set teaches students how ultrasonic sensors work, how to measure distance using sound, how to use thresholds to program the ultrasonic sensor, and how to re-configure the motors and sensor setup window. The lesson is also designed to review how programming loops work.

The Sonic Sojourn Video - This video describes how the ultrasonic sensor works.

The Threshold Reference Page - This handout describes what thresholds are, how they are used in programming, and how to calculate them for the ultrasonic sensor.

Random Numbers Reference Page - This handout describes how to write programs that use random number values using ROBOTC.

The Ultrasonic Sensor PDF Lesson - A printed version of the video designed to complement the video instruction.

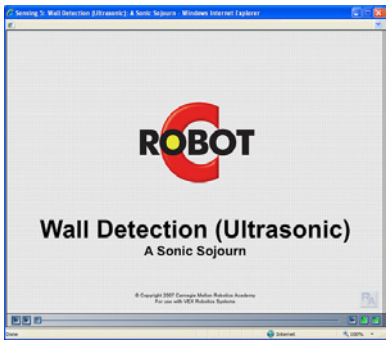
Ultrasonic Programming Challenges - This Lesson Set includes three different programming lessons: The Robo500 Challenge (Level 2), The Table Bot (Level 1) Challenge, and the RoboMower Challenge (Level 2). The challenges are designed so that a teacher can use the same “setup” for multiple challenges. Some challenges get progressively more difficult, while other challenges are designed so that the student completing it is able to quickly solve the same challenge with a different sensor.

Ultrasonic Quiz - Designed to check students understanding of the concept.

© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Ultrasonic Sensor

Sensing - Ultrasonic Sensor Resources



Sensing

Wall Detection A Sonic Sojourn

Robots are precise, reliable, intelligent machines, but only when they are programmed to both sense and respond appropriately. Using a sensor which can only detect an obstacle by contact has drawbacks. You would rather not have to bump into something to know it's there, and neither would your robot.

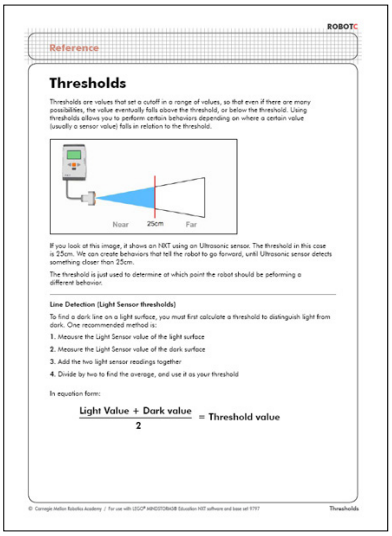
Above right is an Ultrasonic Sensor. Using the same physical principle that a bat or a submarine uses to find its way around, the Ultrasonic Sensor measures distances using sound. It then tells the robot how far away the nearest object in front of it is.

Ultrasonic Sensor detecting a wall (1)
The Ultrasonic Sensor sends out ultrasonic sound waves.

Ultrasonic Sensor detecting a wall (2)
The sound waves hit an obstacle and deflect back. The Ultrasonic Sensor receives the deflected sound waves, then calculates the difference between the time it sent the sound waves and the time it received them. Since the waves travel at a known speed (the speed of sound), the Ultrasonic Sensor can then calculate the distance to the obstacle (in this case, 40 centimeters).

The program you'll write in this lesson will work in a very similar way to the Touch Sensor program you wrote in the previous unit, but instead of using a Touch Sensor to detect obstacles by contact, it will use an Ultrasonic Sensor to detect them at a distance.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Wall Detection / Sonar • 1



Reference

Thresholds

Thresholds are values that set a cutoff in a range of values, so that even if there are many possibilities, the value eventually falls above the threshold, or below the threshold. Using thresholds allows you to perform certain behaviors depending on where a certain value (usually a sensor value) falls in relation to the threshold.

Near — 25cm — Far

If you look at this image, it shows an NXT using an Ultrasonic sensor. The threshold in this case is 25cm. We can create behaviors that tell the robot to go forward, until Ultrasonic sensor detects something closer than 25cm.

The threshold is just used to determine at which point the robot should be peforming a different behavior.

Line Detection (Light Sensor thresholds)

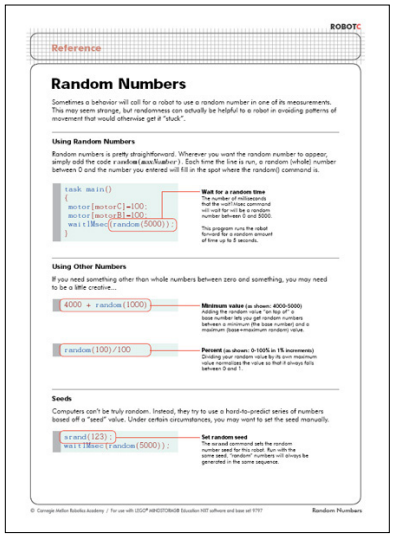
To find a dark line on a light surface, you must first calculate a threshold to distinguish light from dark. One recommended method is:

1. Measure the Light Sensor value of the light surface
2. Measure the Light Sensor value of the dark surface
3. Add the two light sensor readings together
4. Divide by two to find the average, and use it as your threshold

In equation form:

$$\frac{\text{Light Value} + \text{Dark value}}{2} = \text{Threshold value}$$

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Thresholds



Reference

Random Numbers

Sometimes a behavior will call for a robot to use a random number in one of its measurements. This may seem strange, but randomness can actually be helpful to a robot in avoiding patterns of movement that would otherwise get it "stuck".

Using Random Numbers

Random numbers is pretty straightforward. Wherever you want the random number to appear, simply add the code `random(aus/kuf/aur)`. Each time the line is run, a random (whole) number between 0 and the number you entered will fill in the spot where the random() command is.

```
task main()
{
  motor[motorC]=100;
  motor[motorB]=100;
  wait1Msec(random(5000));
}
```

Wait for a random time
The number of milliseconds that the wait1Msec command will wait for will be a random number between 0 and 5000.

This program runs the robot forward for a random amount of time up to 5 seconds.

Using Other Numbers

If you need something other than whole numbers between zero and something, you may need to be a little creative...

```
4000 + random(1000)
```

Minimum value (as shown: 4000-5000)
Adding the random value "on top of" a base number lets you get random numbers between a minimum (the base number) and a maximum (base+maximum random) value.

```
random(100)/100
```

Percent (as shown: 0-100% in 1% increments)
Dividing your random value by its own maximum value normalizes the value so that it always falls between 0 and 1.

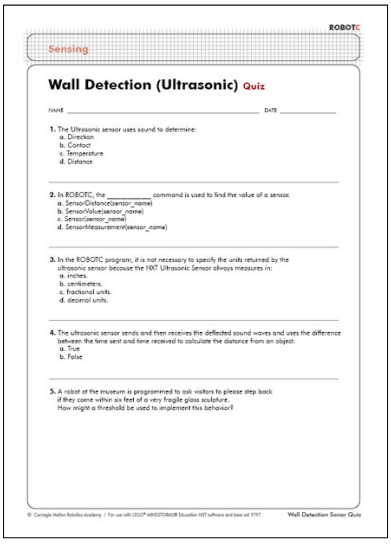
Seeds

Computers can't be truly random. Instead, they try to use a hard-to-predict series of numbers based off a "seed" value. Under certain circumstances, you may want to set the seed manually.

```
srand(123);
wait1Msec(random(5000));
```

Set random seed
The srand() command sets the random number seed for this robot. Run with the same seed, "random" numbers will always be generated in the same sequence.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Random Numbers



Sensing

Wall Detection (Ultrasonic) Quiz

NAME _____ DATE _____

1. The Ultrasonic sensor uses sound to determine:
   a. Direction
   b. Contact
   c. Temperature
   d. Distance

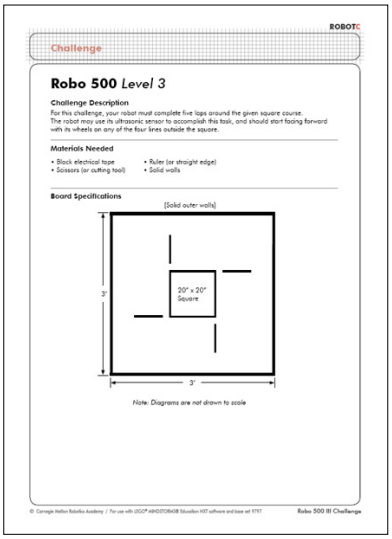
2. In ROBOTC, the _____ command is used to find the value of a sensor.
   a. SensorDistance(sensor_name)
   b. SensorValue(sensor_name)
   c. Sensor(sensor_name)
   d. SensorMeasurement(sensor_name)

3. In the ROBOTC program, it is not necessary to specify the units returned by the ultrasonic sensor because the NXT Ultrasonic Sensor always measures in:
   a. inches.
   b. centimeters.
   c. fractional units.
   d. decimal units.

4. The ultrasonic sensor sends and then receives the deflected sound waves and uses the difference between the time sent and time received to calculate the distance from an object.
   a. True
   b. False

5. A robot at the museum is programmed to ask visitors to please step back if they come within six feet of a very fragile glass sculpture.
   How might a threshold be used to implement this behavior?

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Wall Detection Sonar Quiz



Challenge

Robo 500 *Level 3*

Challenge Description

For this challenge, your robot must complete five laps around the given square course. The robot may use its ultrasonic sensor to accomplish this task, and should start facing forward with its wheels on any of the four lines outside the square.

Materials Needed

- Black electrical tape
- Scissors (or cutting tool)
- Ruler (or straight edge)
- Solid walls

Board Specifications

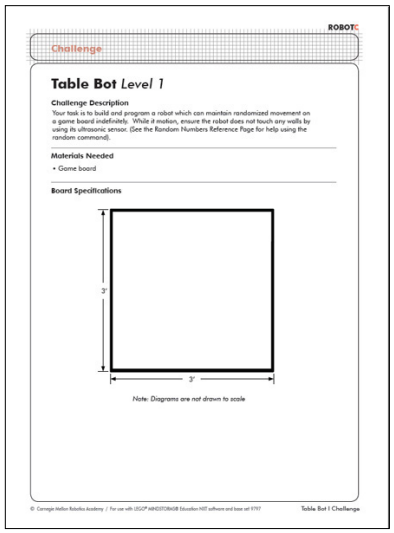
[Solid outer walls]

20" x 20" Square

3' — 3'

Note: Diagrams are not drawn to scale

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Robo 500 III Challenge



Challenge

Table Bot *Level 1*

Challenge Description

Your task is to build and program a robot which can maintain randomized movement on a game board indefinitely. While it motion, ensure the robot does not touch any walls by using its ultrasonic sensor. (See the Random Numbers Reference Page for help using the random command).

Materials Needed

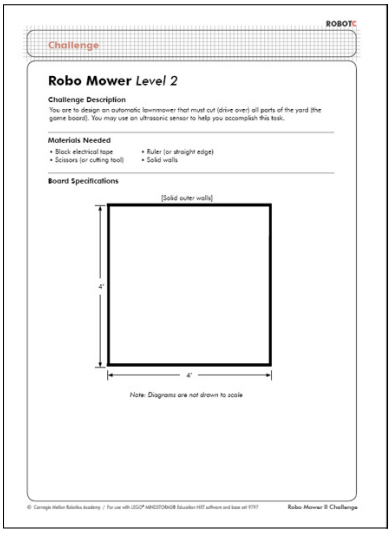
- Game board

Board Specifications

3' — 3'

Note: Diagrams are not drawn to scale

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Table Bot I Challenge



Challenge

Robo Mower *Level 2*

Challenge Description

You are to design an automatic lawnmower that must cut (drive over) all parts of the yard (the game board). You may use an ultrasonic sensor to help you accomplish this task.

Materials Needed

- Black electrical tape
- Scissors (or cutting tool)
- Ruler (or straight edge)
- Solid walls

Board Specifications

[Solid outer walls]

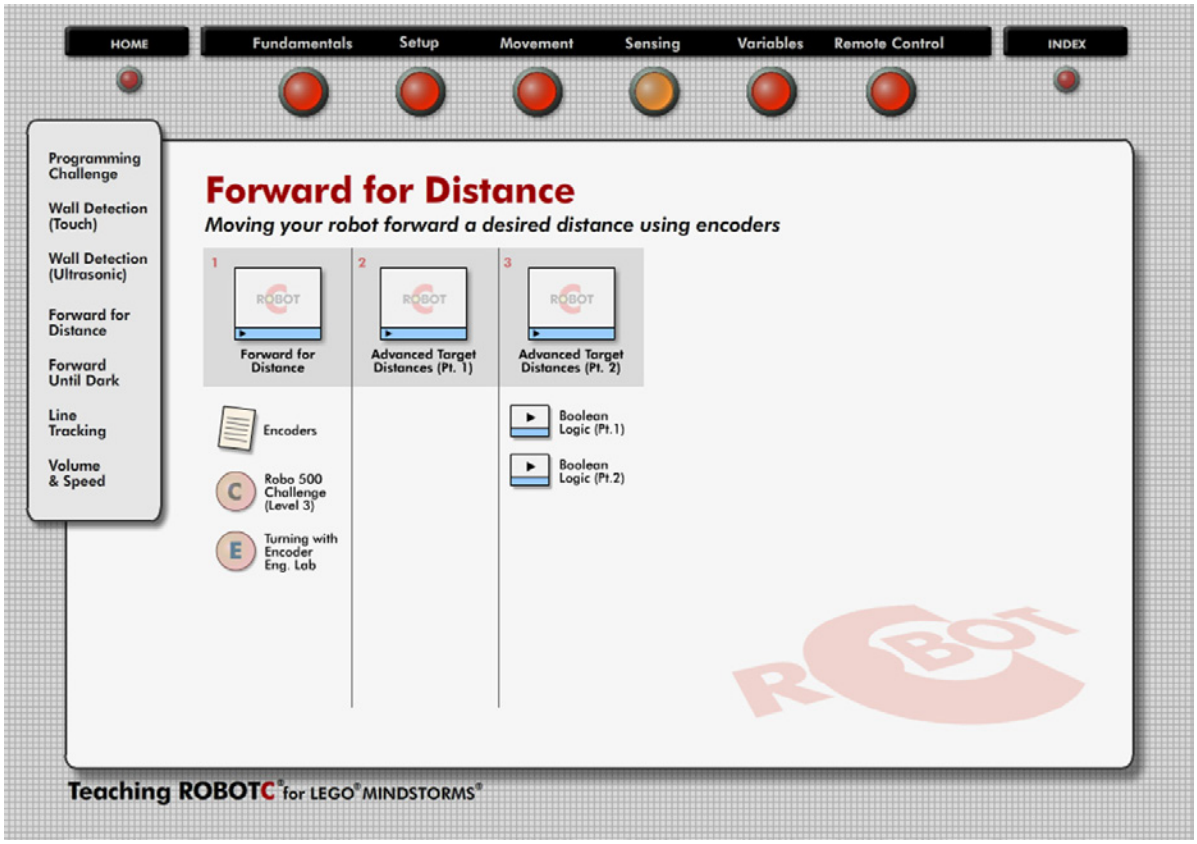
4' — 4'

Note: Diagrams are not drawn to scale

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Robo Mower II Challenge

© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Encoder



Sensing - Encoders

In the Forward for Distance Lesson Set, students learn advanced features of the nMotorEncoder function, allowing them to make very precise movements. The NXT smart motors have encoders built into them that allow the robot to move point-to-point in a very accurate manner. In the first video, students learn a simple method of moving point-to-point. In the second and third video set, students learn how to use the nMotorEncoderTarget function.

Forward for Distance Video - This lesson teaches students about the encoders built into the NXT smart motors, and the nMotorEncoder function.

Advanced Target Distance Part 1 Video - This lesson teaches students about the difference between the nMotorEncoder function and the nMotorEncoderTarget function. It also explains how the nMotorEncoderTarget function is programmed. Students will also learn about a new function that watches the motors “run state”.

Advanced Target Distance Part 2 Video - This lesson shows students how to write the nMotorEncoderTarget program.

Boolean Logic Videos - Comparison operators and logical operators are key elements that all students must know. This video set is included at several spots throughout the curriculum for student review.

Encoders Reference Page - The encoders reference page is a handout that can be used as a reference or study guide.

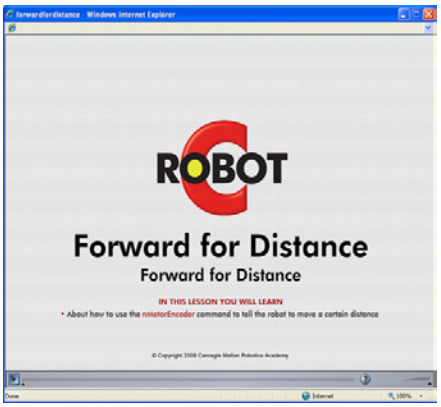
Turning with Encoder Engineering Lab - The Encoder Engineering Lab has students calculate encoder counts to turn specific angles. This data will help them with all programming moving forward.

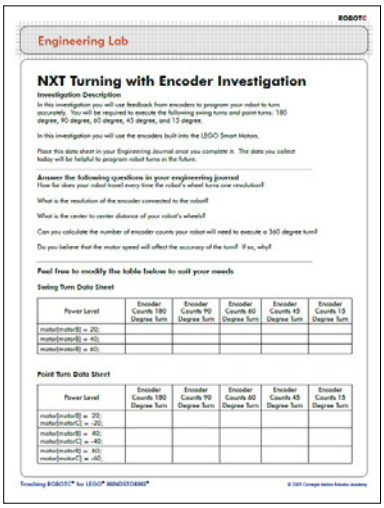
The Robo500 (Level3) Programming Challenge - Students have seen this challenge before. Now they will be able to apply what they’ve learned in the encoder lessons to solve the problem more perfectly.

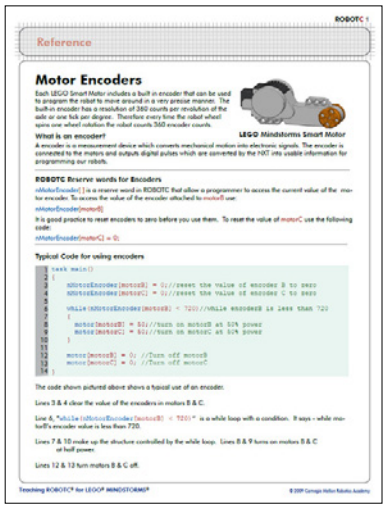
© 2009 Carnegie Mellon Robotics Academy

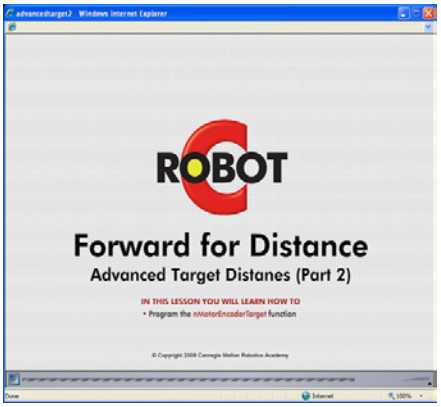
Introduction to Programming/Sensing/Encoder

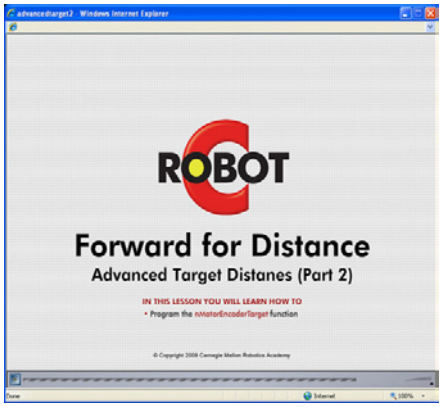
Sensing - Encoder Resources

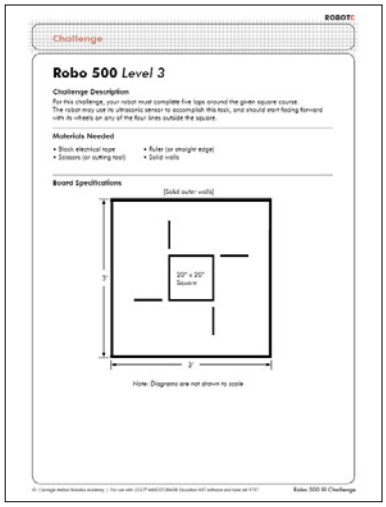


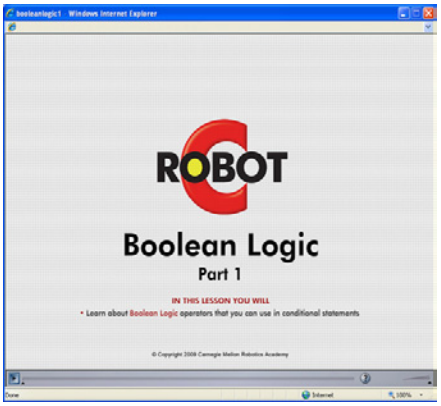


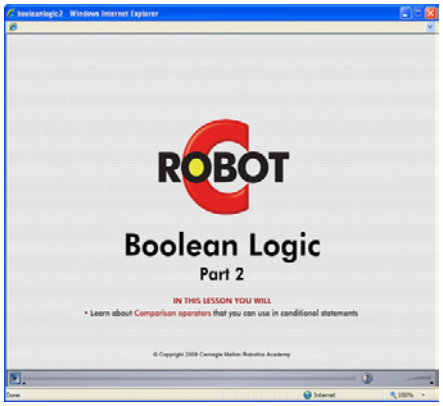






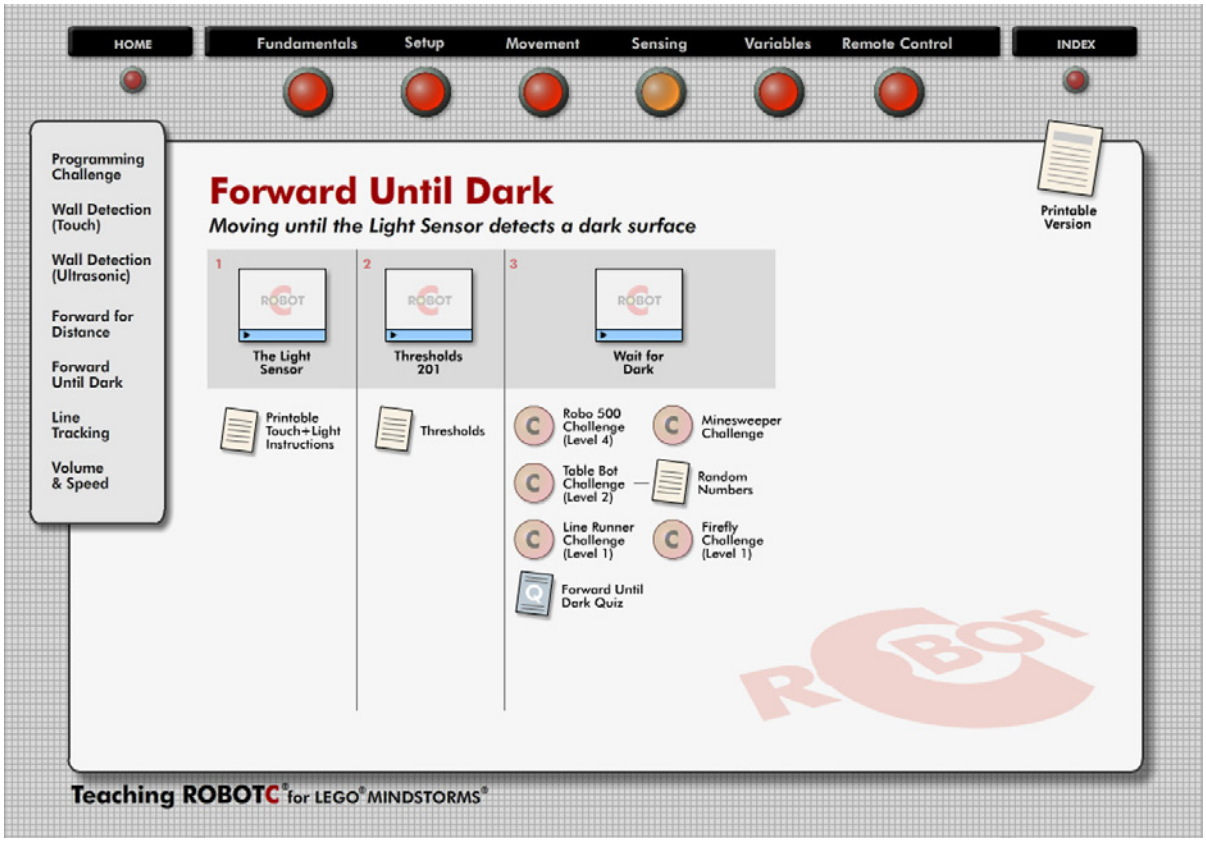






© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Light Sensor



Sensing - Light Sensor

The Forward Until Dark Lesson Set introduces students to how the light sensor works, how to troubleshoot programs that use the light sensor, and how to calculate threshold values for light sensors. The Lesson Set has the following resources:

The Light Sensor Video - The light sensor video reviews how the touch and ultrasonic sensors work and then describes how the LEGO light sensor works. The video also describes things to consider to optimize how the sensor works.

The Thresholds 201 Video - This lesson describes how to calculate thresholds for the light sensor. It also demonstrates the physical steps that a programmer needs to do to calculate a light sensor threshold value.

The Wait for Dark Video - This lesson begins with a prior program, then adapts the program to use the light sensor. Students will learn how to configure the Motors and Setup Configuration menu for the light sensor, and then are given directions that allow them to complete the program.

The Threshold and Random Numbers Reference Pages - These handouts are the same handouts used in the previous lesson and are useful to students still learning these foundational programming concepts.

The Forward Until Dark Printable PDF - This document is a printed version of the three video lessons in the Forward Until Dark Lesson Set.

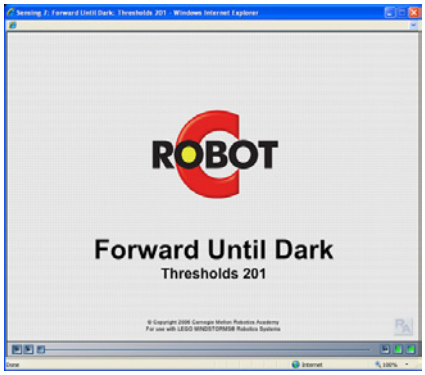
The Programming Challenges - There are five programming challenges with this lesson: The Robo500 Challenge (Level4), the Table Bot Challenge (Level 2), the Line Runner Challenge (Level1), the Minesweeper Challenge, and the Firefly Challenge (Level1). All students may not complete all programming challenges.

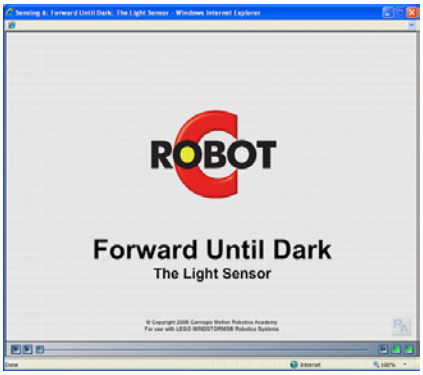
Forward Until Dark Lesson Set Quiz - A quiz to check student's understanding.

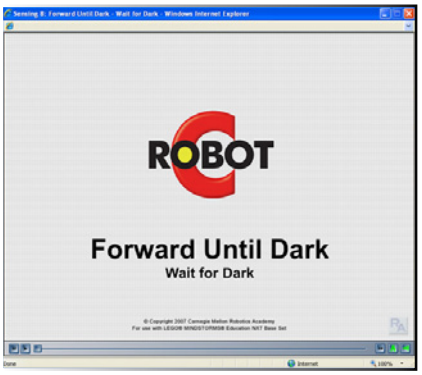
© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Light Sensor

Sensing - Light Sensor Resources







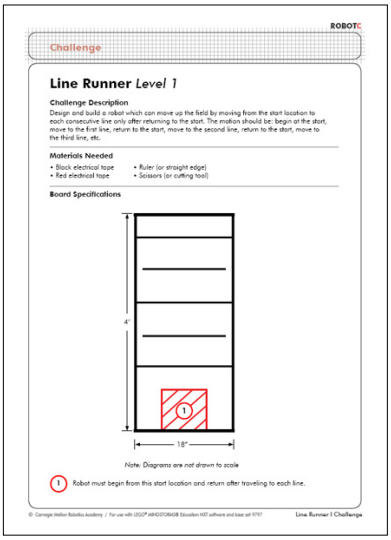
Sensing

Forward Until Dark Light Sensor

In this lesson, you will learn how the Light Sensor works, and how its feedback compares to the Touch and Ultrasonic Sonar sensors.

Touch Sensor
The Touch Sensor detects physical contact with the orange trigger, and returns a SensorValue of 1 if it is pressed in, or 0 if it is not.

Ultrasonic Sensor
The Ultrasonic (sometimes called Sonar) Sensor sends out pulses of sound and measures the time it takes for the sound waves to bounce off an object and return. Since the speed of sound is known, the sensor calculates the distance based on the time, and reports the distance in centimeters as its SensorValue



Challenge

Line Runner *Level 1*

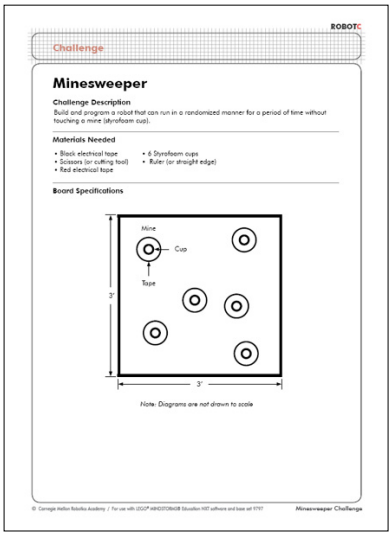
Challenge Description
Design and build a robot which can move up the field by moving from the start location to each consecutive line only after returning to the start. The motion should be: begin at the start, move to the first line, return to the start, move to the second line, return to the start, move to the third line, etc.

Materials Needed
- Black electrical tape
- Red electrical tape
- Ruler (or straight edge)
- Scissors (or cutting tool)

Board Specifications

Note: Diagrams are not drawn to scale

1 Robot must begin from this start location and return after traveling to each line.



Challenge

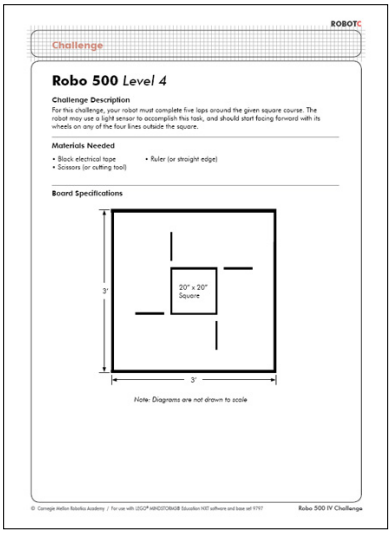
Minesweeper

Challenge Description
Build and program a robot that can run in a randomized manner for a period of time without touching a mine (styrofoam cup).

Materials Needed
- Black electrical tape
- Scissors (or cutting tool)
- Red electrical tape
- 6 Styrofoam cups
- Ruler (or straight edge)

Board Specifications

Note: Diagrams are not drawn to scale



Challenge

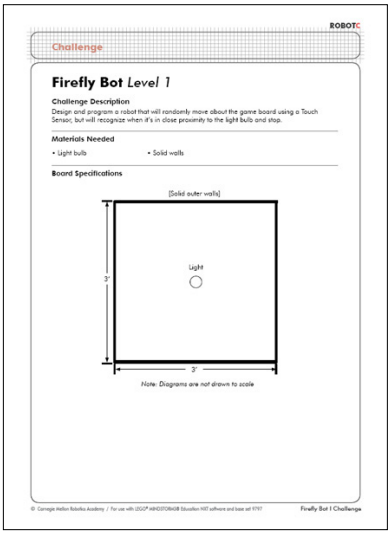
Robo 500 *Level 4*

Challenge Description
For this challenge, your robot must complete five laps around the given square course. The robot may use a light sensor to accomplish this task, and should start facing forward with its wheels on any of the four lines outside the square.

Materials Needed
- Black electrical tape
- Scissors (or cutting tool)
- Ruler (or straight edge)

Board Specifications

Note: Diagrams are not drawn to scale



Challenge

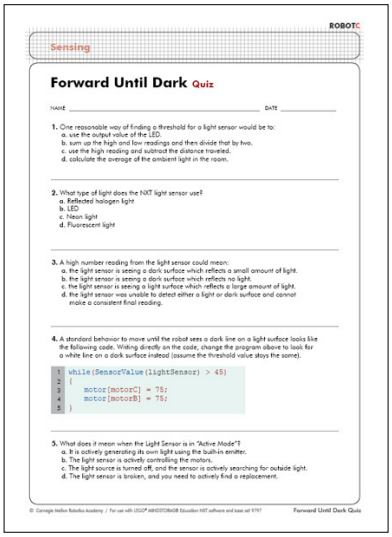
Firefly Bot *Level 1*

Challenge Description
Design and program a robot that will randomly move about the game board using a Touch Sensor, but will recognize when it's in close proximity to the light bulb and stop.

Materials Needed
- Light bulb
- Solid walls

Board Specifications

Note: Diagrams are not drawn to scale



Sensing

Forward Until Dark Quiz

NAME _____ DATE _____

1. One reasonable way of finding a threshold for a light sensor would be to:
   a. use the output value of the LED.
   b. sum up the high and low readings and then divide that by two.
   c. use the high reading and subtract the distance traveled.
   d. calculate the average of the ambient light in the room.

2. What type of light does the NXT light sensor use?
   a. Reflected halogen light
   b. LED
   c. Neon light
   d. Fluorescent light

3. A high number reading from the light sensor could mean:
   a. the light sensor is seeing a dark surface which reflects a small amount of light.
   b. the light sensor is seeing a dark surface which reflects no light.
   c. the light sensor is seeing a light surface which reflects a large amount of light.
   d. the light sensor was unable to detect either a light or dark surface and cannot make a consistent final reading.

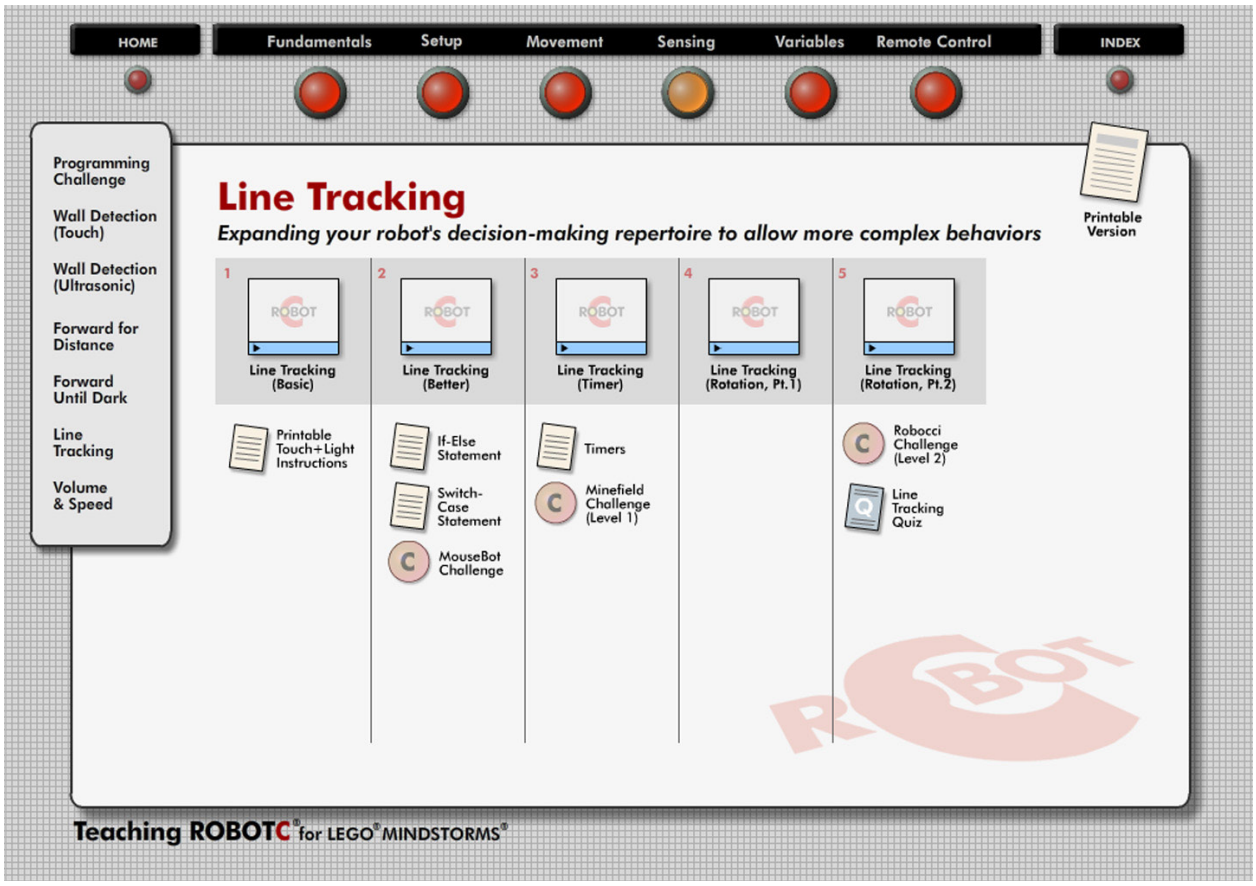
4. A standard behavior to move until the robot sees a dark line on a light surface looks like the following code. Working directly on the code, change the program above to look for a white line on a dark surface instead (assume the threshold value stays the same).

```
while(SensorValue(lightSensor) > 45)
{
    motor[motorC] = 75;
    motor[motorB] = 75;
}
```

5. What does it mean when the Light Sensor is in "Active Mode"?
   a. It is actively generating its own light using the built-in emitter.
   b. The light sensor is actively controlling the motors.
   c. The light source is turned off, and the sensor is actively searching for outside light.
   d. The light sensor is broken, and you need to actively find a replacement.

© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Light Sensor/LineTracking



Sensing - Light Sensor LineTracking

The Line Tracking Lesson Set breaks line tracking, a seemingly complex behavior, into a set of simple robot behaviors. It introduces students to a new structure, the if-else structure. Students learn to use multiple sensor feedback when they first use feedback from internal timers and light sensors to control how far their robot moves. Once they solve that problem, they then use feedback from the NXT smart motors and the light sensor to determine how far their robot travels. At the end of this Lesson Set, students will have all of the tools that they need to complete the obstacle course challenge. This Lesson Set uses the following video resources:

The Line Tracking (Basic) Video - This video teaches students how line tracking works, and how to break complex behaviors into simple behaviors. It also introduces the students to a naive way to solve the line track programming problem using while loops. This solution works, but has serious limitations.

Line Tracking (Better) Video - This lesson teaches students how to use feedback from multiple sensors with their robot. They discover that their naive solution using only “while loops” doesn’t always work. The video explains if/else structures, explaining the code as it talks students through the solution.

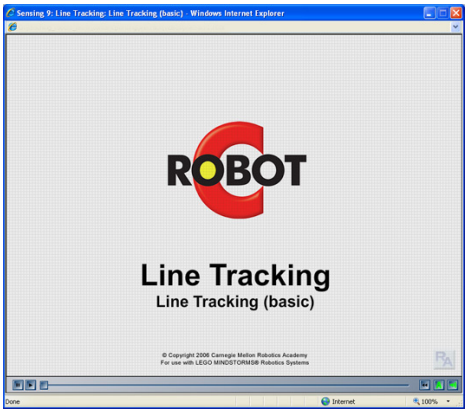
The Line Track Timer Video - This video teaches students how to use the internal timers built into the NXT to control how long the robot does something. This is a very valuable tool that the student will be able to use in many other situations.

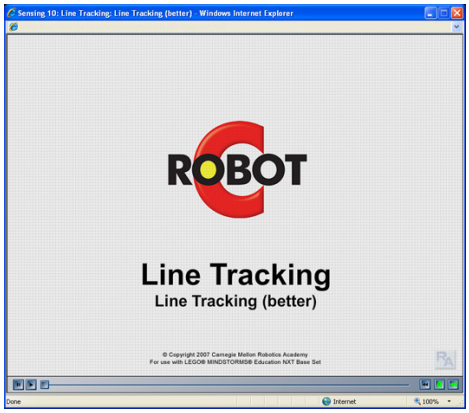
The Line Tracking Rotation Parts 1 & 2 Videos - Teaches the student to use feedback from the encoders built into the NXT smart motors to determine when to stop tracking a line.

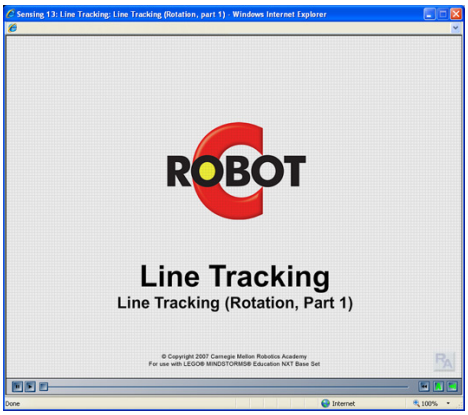
© 2009 Carnegie Mellon Robotics Academy

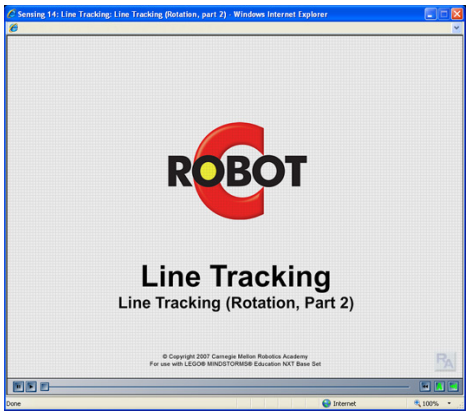
Introduction to Programming/Sensing/Light Sensor/LineTracking

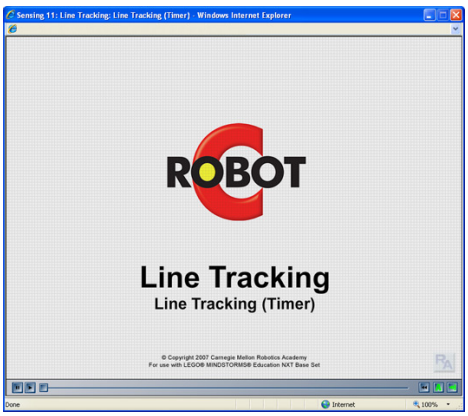
Sensing - Light Sensor LineTracking Video Resources











Sensing - Light Sensor LineTracking Print Resources

If-Else Reference Page - Students will use the if-else structure from this point forward in their programming career. This is a one page reference sheet that shows them how it works.

Switch Case Statement - Another option for programmers that have to control multiple conditions is the switch case. This resource is a multi-page handout that students can reference when using switch cases.

Timer Reference Page - The timer reference page demonstrates specific reserved words used in ROBOTC to control the internal timers built into the NXT.

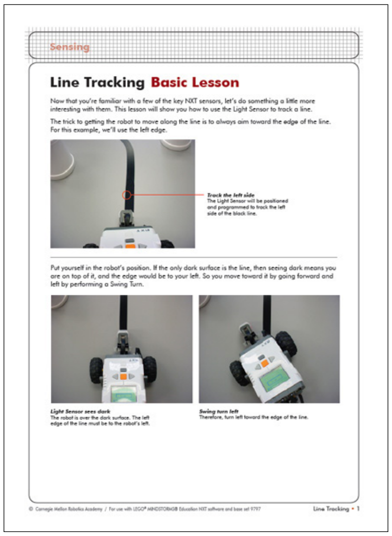
Line Tracking Programming Challenges - This Lesson Set includes three open-ended challenges designed to support the lesson: the MouseBot Challenge, the Minefield Challenge (Level1), and the Robocci Challenge (Level2). All students may not complete every programming challenge.

Line Track Quiz - The line track quiz is included to check student's understanding.

© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Light Sensor/LineTracking

Sensing - Light Sensor LineTracking Print Resources



Sensing

Line Tracking Basic Lesson

Now that you're familiar with a few of the key NXT sensors, let's do something a little more interesting with them. This lesson will show you how to use the Light Sensor to track a line.

The trick to getting the robot to move along the line is to always aim toward the edge of the line. For this example, we'll use the left edge.

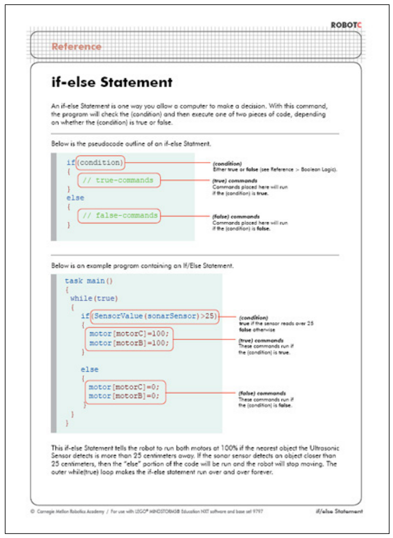
Track the left side
The Light Sensor will be positioned and programmed to track the left side of the black line.

Put yourself in the robot's position. If the only dark surface is the line, then seeing dark means you are on top of it, and the edge would be to your left. So you move toward it by going forward and left by performing a Swing Turn.

Light Sensor sees dark
The robot is over the dark surface. The left edge of the line must be to the robot's left.

Swing turn left
Therefore, turn left toward the edge of the line.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Line Tracking • 1



ROBOTC

Reference

if-else Statement

An if-else Statement is one way you allow a computer to make a decision. With this command, the program will check the (condition) and then execute one of two pieces of code, depending on whether the (condition) is true or false.

Below is the pseudocode outline of an if-else Statment.

```
if(condition)
{
  // true-commands
}
else
{
  // false-commands
}
```

(condition) Either true or false (see Reference > Boolean Logic).
(true) commands Commands placed here will run if the (condition) is true.
(false) commands Commands placed here will run if the (condition) is false.

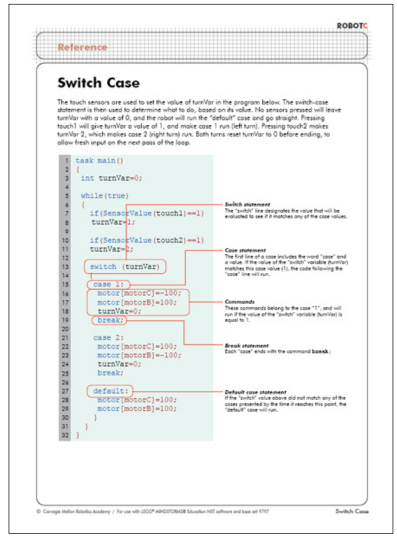
Below is an example program containing an If/Else Statement.

```
task main()
{
  while(true)
  {
    if(SensorValue(sonarSensor)>25)
    {
      motor[motorC]=100;
      motor[motorB]=100;
    }
    else
    {
      motor[motorC]=0;
      motor[motorB]=0;
    }
  }
}
```

(condition) true if the sensor reads over 25, false otherwise
(true) commands These commands run if the (condition) is true.
(false) commands These commands run if the (condition) is false.

This if-else Statement tells the robot to run both motors at 100% if the nearest object the Ultrasonic Sensor detects is more than 25 centimeters away. If the sonar sensor detects an object closer than 25 centimeters, then the "else" portion of the code will be run and the robot will stop moving. The outer while(true) loop makes the if-else statement run over and over forever.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — if/else Statement



ROBOTC

Reference

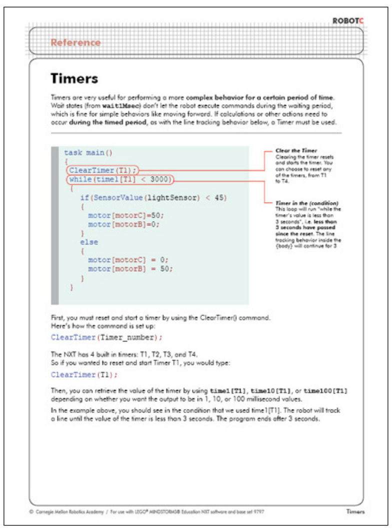
Switch Case

The touch sensors are used to set the value of turnVar in the program below. The switch-case statement is then used to determine what to do, based on its value. No sensors pressed will leave turnVar with a value of 0, and the robot will run the "default" case and go straight. Pressing touch1 will give turnVar a value of 1, and make case 1 run (left turn). Pressing touch2 makes turnVar 2, which makes case 2 (right turn) run. Both turns reset turnVar to 0 before ending, to allow fresh input on the next pass of the loop.

```
task main()
{
  int turnVar=0;

  while(true)
  {
    if(SensorValue(touch1)==1)
    turnVar=1;

    if(SensorValue(touch2)==1)
    turnVar=2;

    switch (turnVar)
    {
      case 1:
        motor[motorC]=-100;
        motor[motorB]=100;
        turnVar=0;
        break;

      case 2:
        motor[motorC]=100;
        motor[motorB]=-100;
        turnVar=0;
        break;

      default:
        motor[motorC]=100;
        motor[motorB]=100;
    }
  }
}
```

Switch statement The "switch" line designates the value that will be evaluated to see if it matches any of the case values.
Case statement The first line of a case includes the word "case" and a value. If the value of the "switch" variable (turnVar) matches this case value (1), the code following the "case" line will run.
Commands These commands belong to the case "1", and will run if the value of the "switch" variable (turnVar) is equal to 1.
Break statement Each "case" ends with the command **break**;
Default case statement If the "switch" value above did not match any of the cases presented by the time it reaches this point, the "default" case will run.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Switch Case



ROBOTC

Reference

Timers

Timers are very useful for performing a more **complex behavior for a certain period of time**. Wait states (from **wait1Msec**) don't let the robot execute commands during the waiting period, which is fine for simple behaviors like moving forward. If calculations or other actions need to occur **during the timed period**, as with the line tracking behavior below, a Timer must be used.

```
task main()
{
  ClearTimer(T1);
  while(time1[T1] < 3000)
  {
    if(SensorValue(lightSensor) < 45)
    {
      motor[motorC]=50;
      motor[motorB]=0;
    }
    else
    {
      motor[motorC] = 0;
      motor[motorB] = 50;
    }
  }
}
```

Clear the Timer Clearing the timer resets and starts the timer. You can choose to reset any of the timers, from T1 to T4.
Timer in the (condition) This loop will run "while the timer's value is less than 3 seconds", i.e. **less than 3 seconds have passed since the reset**. The line tracking behavior inside the {body} will continue for 3

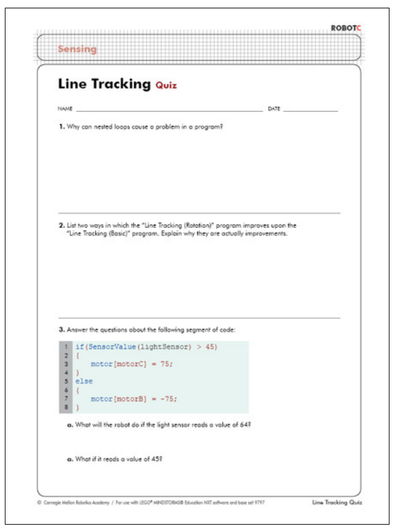
First, you must reset and start a timer by using the ClearTimer() command.
Here's how the command is set up:
`ClearTimer(Timer_number);`

The NXT has 4 built in timers: T1, T2, T3, and T4.
So if you wanted to reset and start Timer T1, you would type:
`ClearTimer(T1);`

Then, you can retrieve the value of the timer by using **time1[T1]**, **time10[T1]**, or **time100[T1]** depending on whether you want the output to be in 1, 10, or 100 millisecond values.

In the example above, you should see in the condition that we used time1[T1]. The robot will track a line until the value of the timer is less than 3 seconds. The program ends after 3 seconds.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Timers



ROBOTC

Sensing

Line Tracking Quiz

NAME ____________ DATE ________

1. Why can nested loops cause a problem in a program?

2. List two ways in which the "Line Tracking (Rotation)" program improves upon the "Line Tracking (Basic)" program. Explain why they are actually improvements.

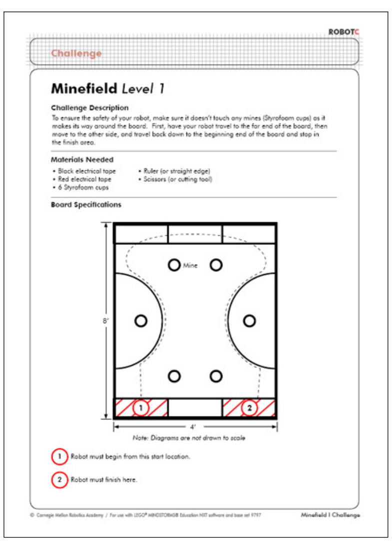
3. Answer the questions about the following segment of code:

```
if(SensorValue(lightSensor) > 45)
{
  motor[motorC] = 75;
}
else
{
  motor[motorB] = -75;
}
```

a. What will the robot do if the light sensor reads a value of 64?

a. What if it reads a value of 45?

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Line Tracking Quiz



ROBOTC

Challenge

Minefield *Level 1*

Challenge Description
To ensure the safety of your robot, make sure it doesn't touch any mines (Styrofoam cups) as it makes its way around the board. First, have your robot travel to the far end of the board, then move to the other side, and travel back down to the beginning end of the board and stop in the finish area.

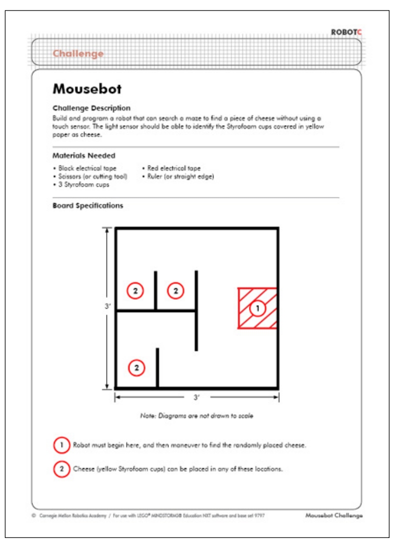
Materials Needed
- Black electrical tape
- Red electrical tape
- 6 Styrofoam cups
- Ruler (or straight edge)
- Scissors (or cutting tool)

Board Specifications

Note: Diagrams are not drawn to scale

(1) Robot must begin from this start location.
(2) Robot must finish here.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Minefield I Challenge



ROBOTC

Challenge

Mousebot

Challenge Description
Build and program a robot that can search a maze to find a piece of cheese without using a touch sensor. The light sensor should be able to identify the Styrofoam cups covered in yellow paper as cheese.

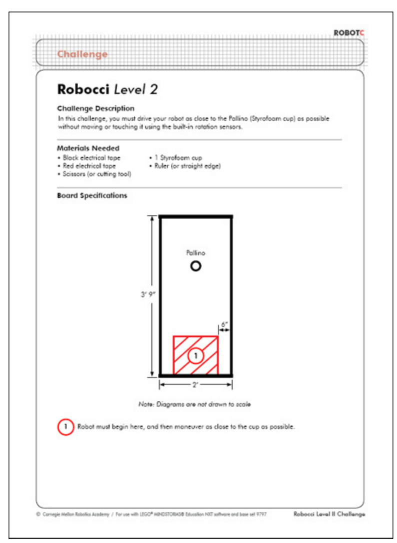
Materials Needed
- Black electrical tape
- Scissors (or cutting tool)
- 3 Styrofoam cups
- Red electrical tape
- Ruler (or straight edge)

Board Specifications

Note: Diagrams are not drawn to scale

(1) Robot must begin here, and then maneuver to find the randomly placed cheese.
(2) Cheese (yellow Styrofoam cups) can be placed in any of these locations.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Mousebot Challenge



ROBOTC

Challenge

Robocci *Level 2*

Challenge Description
In this challenge, you must drive your robot as close to the Pallino (Styrofoam cup) as possible without moving or touching it using the built-in rotation sensors.

Materials Needed
- Black electrical tape
- Red electrical tape
- Scissors (or cutting tool)
- 1 Styrofoam cup
- Ruler (or straight edge)

Board Specifications

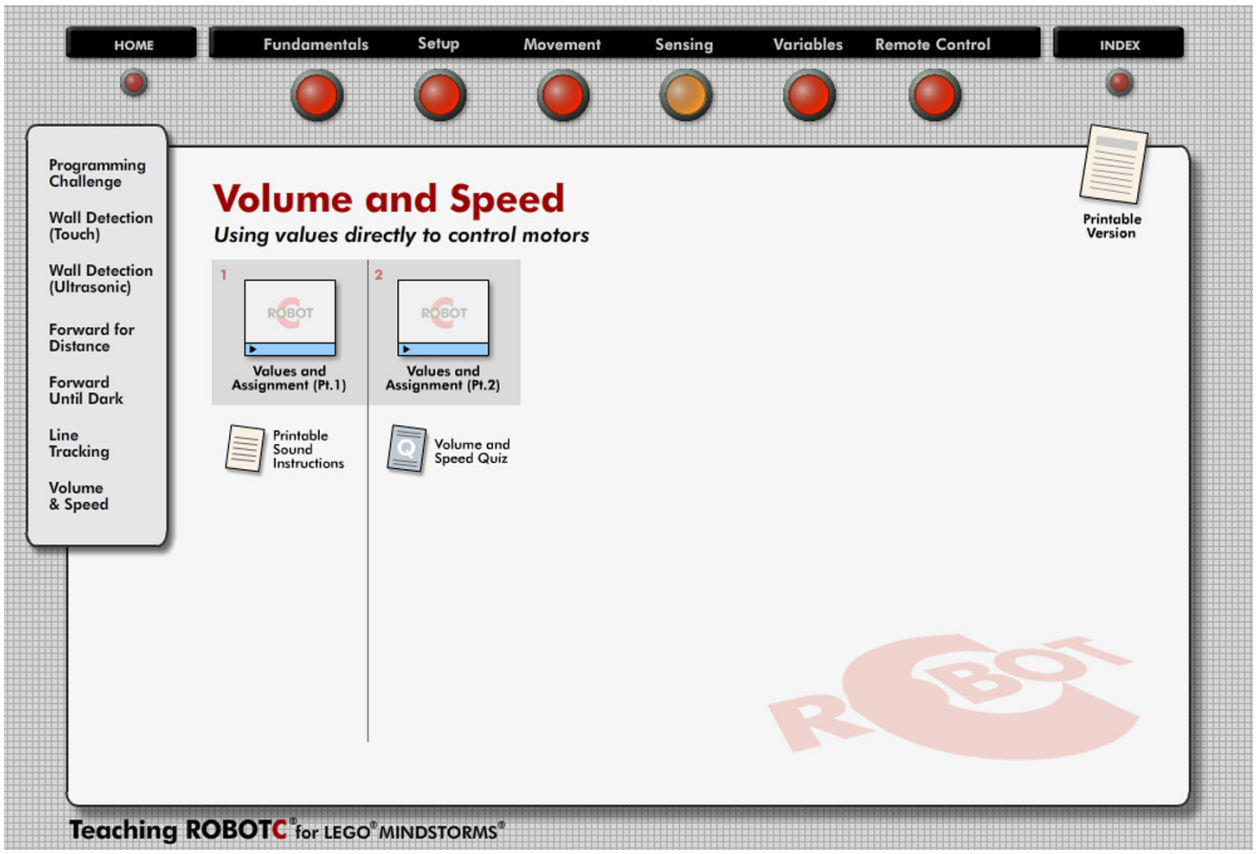
Note: Diagrams are not drawn to scale

(1) Robot must begin here, and then maneuver as close to the cup as possible.

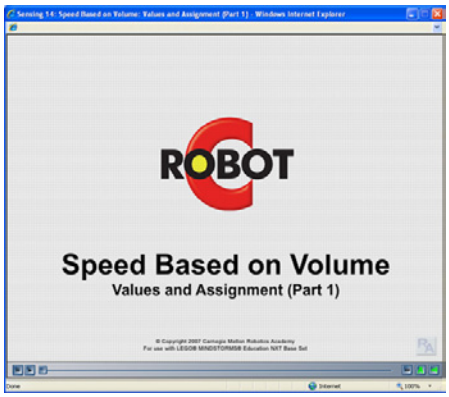
© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Robocci Level II Challenge

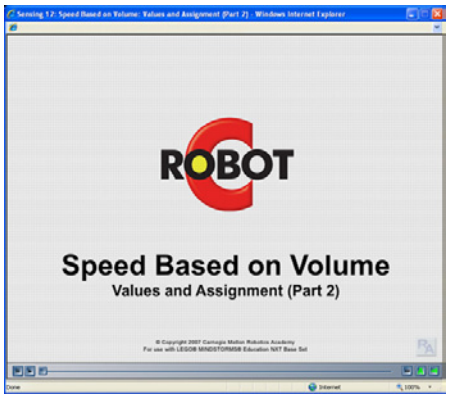
© 2009 Carnegie Mellon Robotics Academy

Introduction to Programming/Sensing/Sound Sensor

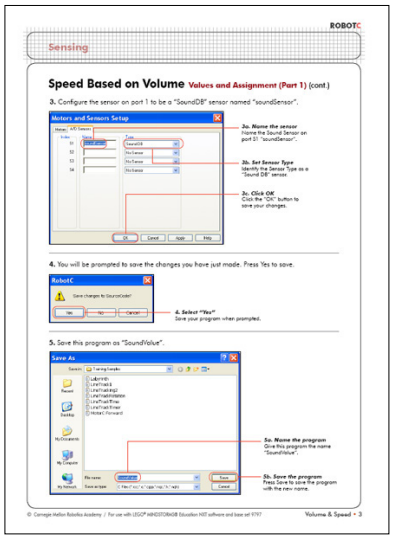


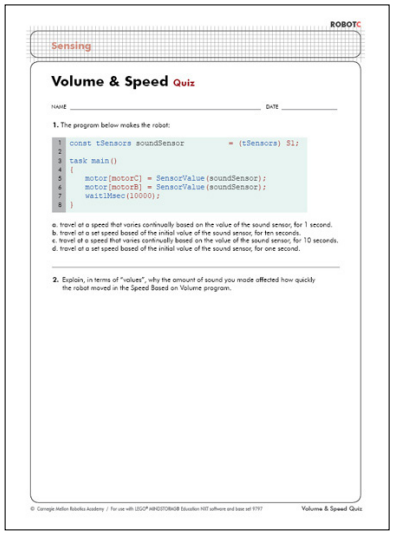
Sensing - Sound Sensor





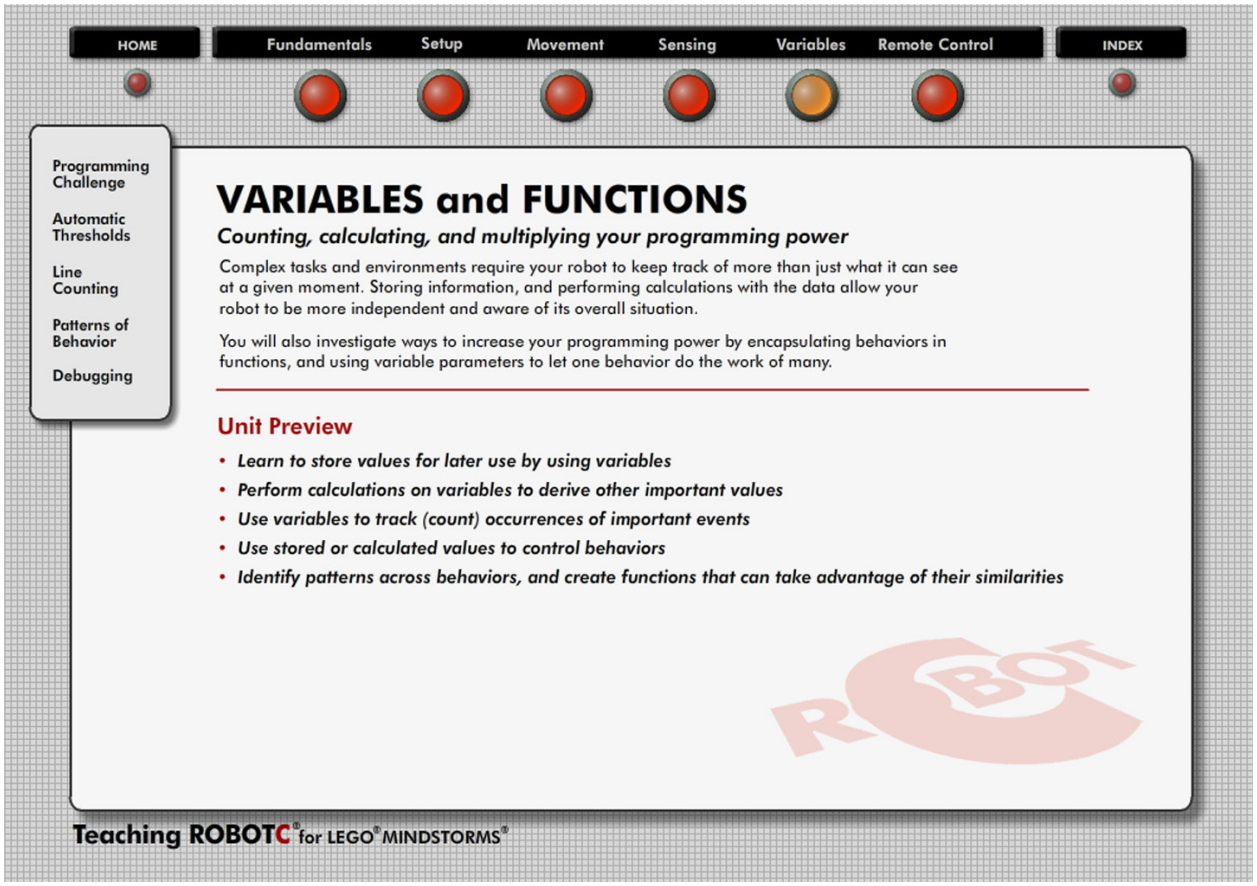
In the Speed Based on Sound Lesson Set, students are challenged to map the values of the sound sensor to the motor powers so that they can control the motors speed based on sound. This lesson reinforces the idea that a programmer needs to continually update the sensor value if they want to use the sensor value to control something real time. This is a very important concept that beginning programmers often miss. In this lesson, students also practice using the internal timers and the reserved words that are associated with it. The Lesson Set includes: two videos, a 9 page PDF that aligns with the video, and a quiz.





© 2009 Carnegie Mellon Robotics Academy

Programming/Sensing/Variables and Functions



Sensing - Variables and Functions

The Variables and Functions Unit contains four Lesson Sets: Automatic Thresholds, Line Counting, Patterns of Behaviors and Debugging. Each of these Lesson Sets teaches a major programming concepts. Automatic Thresholds introduces the concept of variables, variable types, how to name variables, and how to manipulate them. The Line Counting Lesson Set builds on the variable concept and teaches students how they can manipulate the value of variables. Patterns of Behaviors introduces functions, how to pass parameters in functions, and how functions can be used to simplify programs. And finally, the Debugging Lesson Set, offers two lessons on how to use ROBOTC's debugger to troubleshoot programs.

The previous programming units, Movement and Sensing, introduce basic robot programming concepts. These lessons begin to introduce basic programming concepts!

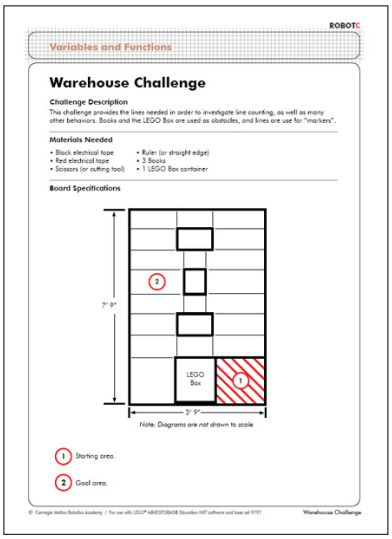
© 2009 Carnegie Mellon Robotics Academy

Programming/Variables and Functions/Programming Challenge

Sensing - Variables and Functions/Programming Challenge

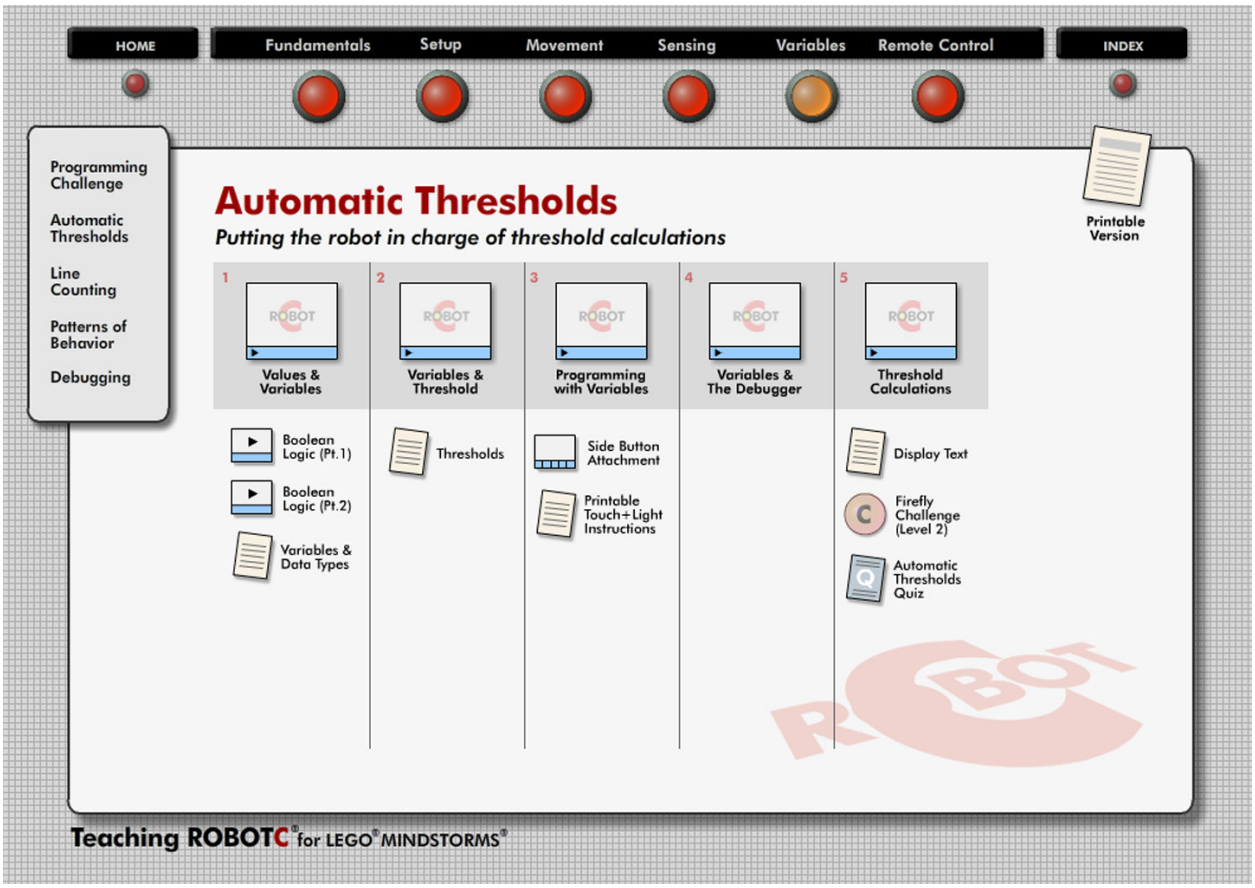
The Warehouse Programming Challenge asks students to developed a customizable program that allows them to program their robot to move to any spot in the warehouse. As students move through the Automatic Thresholds, Line Counting and Patterns of Behavior Lesson Sets they will gain the skills needed to solve the programming challenge.

There are two resources available to introduce the programming challenge: ***the Warehouse Challenge PDF*** and ***the Warehouse Challenge Solution Video***.



© 2009 Carnegie Mellon Robotics Academy

Programming/Variables and Functions/Automatic Thresholds



Sensing - Variables and Functions/Automatic Thresholds

The Automatic Thresholds Lesson Set introduces a variables; a very large concept in computer programming. The step-by-step guided lesson will allow students to solve this programming problem, but the concept will need to be revisited multiple times before students become competent using variables.

Values & Variable Video - This video introduces variables, variable types, and how to name variables.

Variables and Thresholds Video - This video reviews how to calculate thresholds for the light sensor, the programming problem that students will solve in this Lesson Set, and defines the initial variables used to solve this problem.

Programming with Variables Video - This video walks students through the automatic threshold calculation problem beginning with configuring the light and touch sensor and continuing through the initial setup of the problem.

Variables and the Debugger Video - This video shows students how to use ROBOTC's debugger to find the problem in their initial programming solution. They will discover how fast computer processors work, and be shown a solution.

Threshold Calculations Video - This video shows students how to do calculations using variables and how to display values to the NXT display screen.

The Boolean Logic Video Set - Students have seen this set in an earlier lesson; this is a good time to review.

Variable and Data Types Reference - This is a multiple page handout that students can use as a study guide.

Threshold Values Reference - One page reference guide that students can use to review threshold values.

Display Text Reference - This multiple page lesson shows students step-by-step how to display text on their screens.

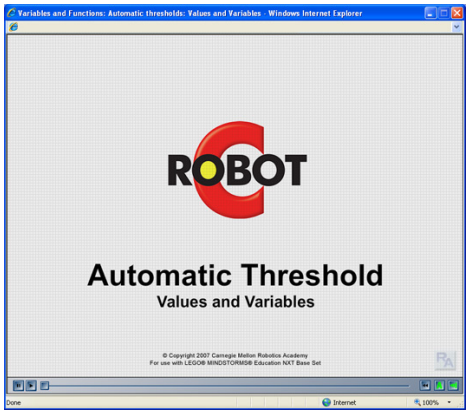
Firefly Challenge (Level2) - This challenge requires students to find a light value and report the value to their screen.

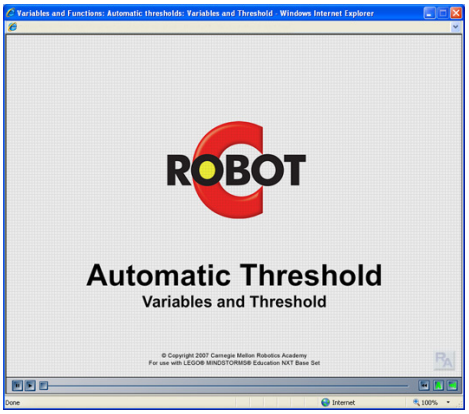
Automatics Thresholds PDF - A 36 page PDF that covers what was taught in the video lessons in a print format.

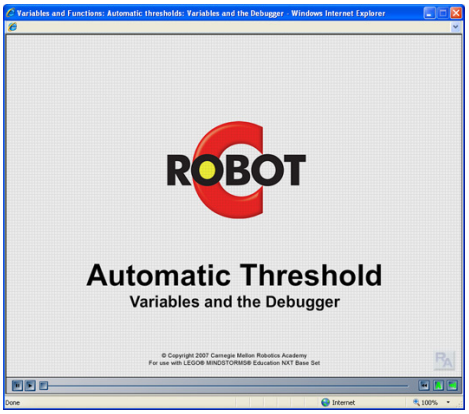
© 2009 Carnegie Mellon Robotics Academy

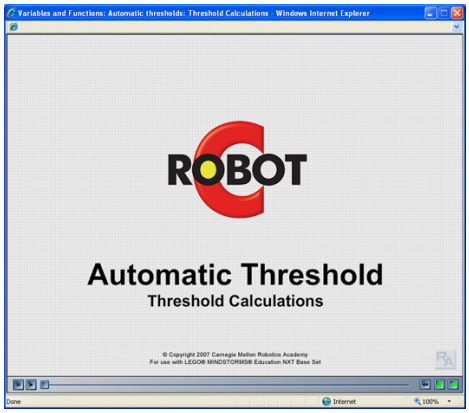
Programming/Variables and Functions/Automatic Thresholds

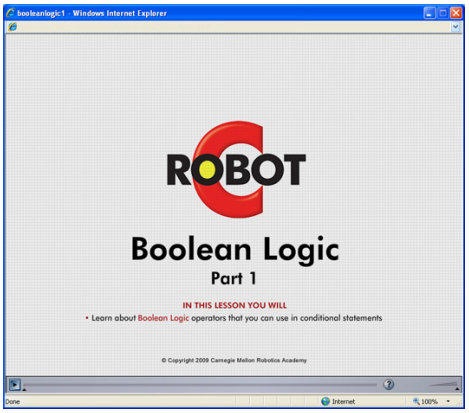
Sensing - Variables and Functions/Automatic Thresholds - Video Resources

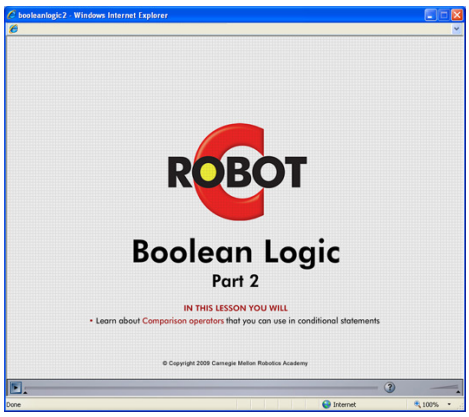








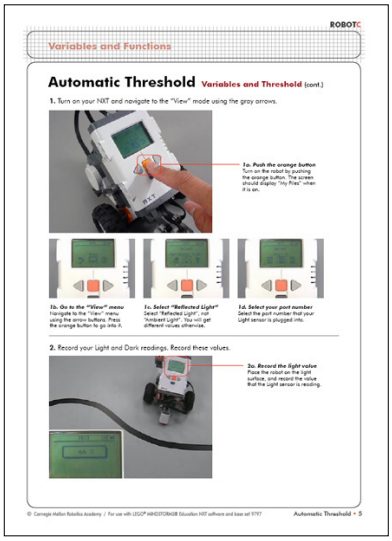


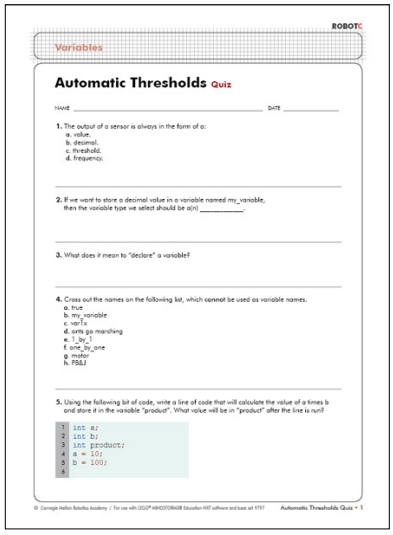


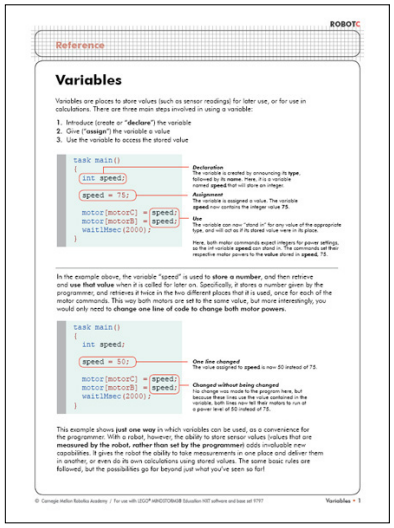
© 2009 Carnegie Mellon Robotics Academy

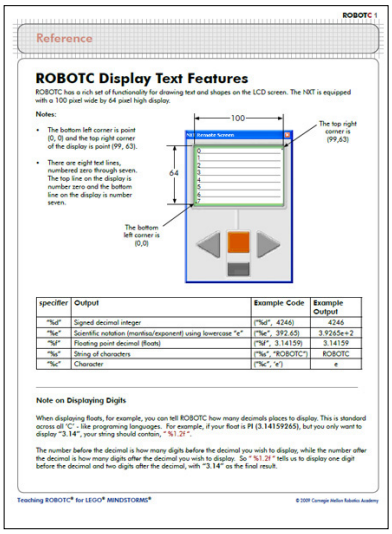
Programming/Variables and Functions/Automatic Thresholds

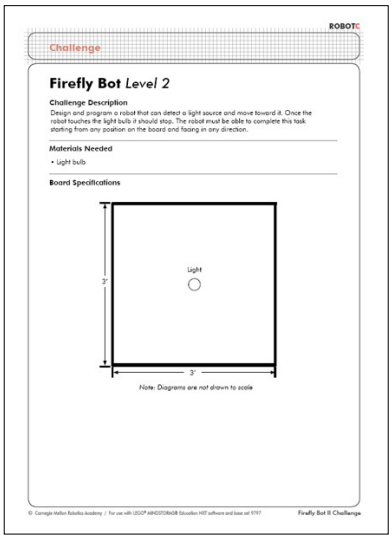
Sensing - Variables and Functions/Automatic Thresholds - Print Resources

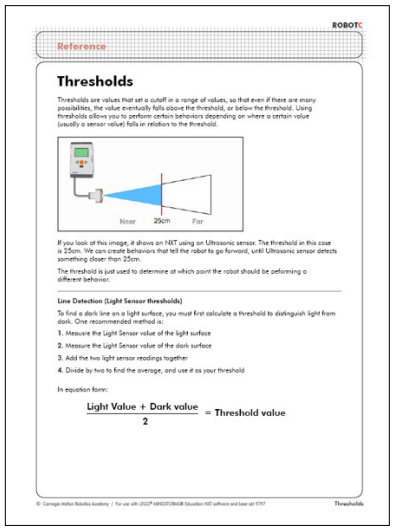






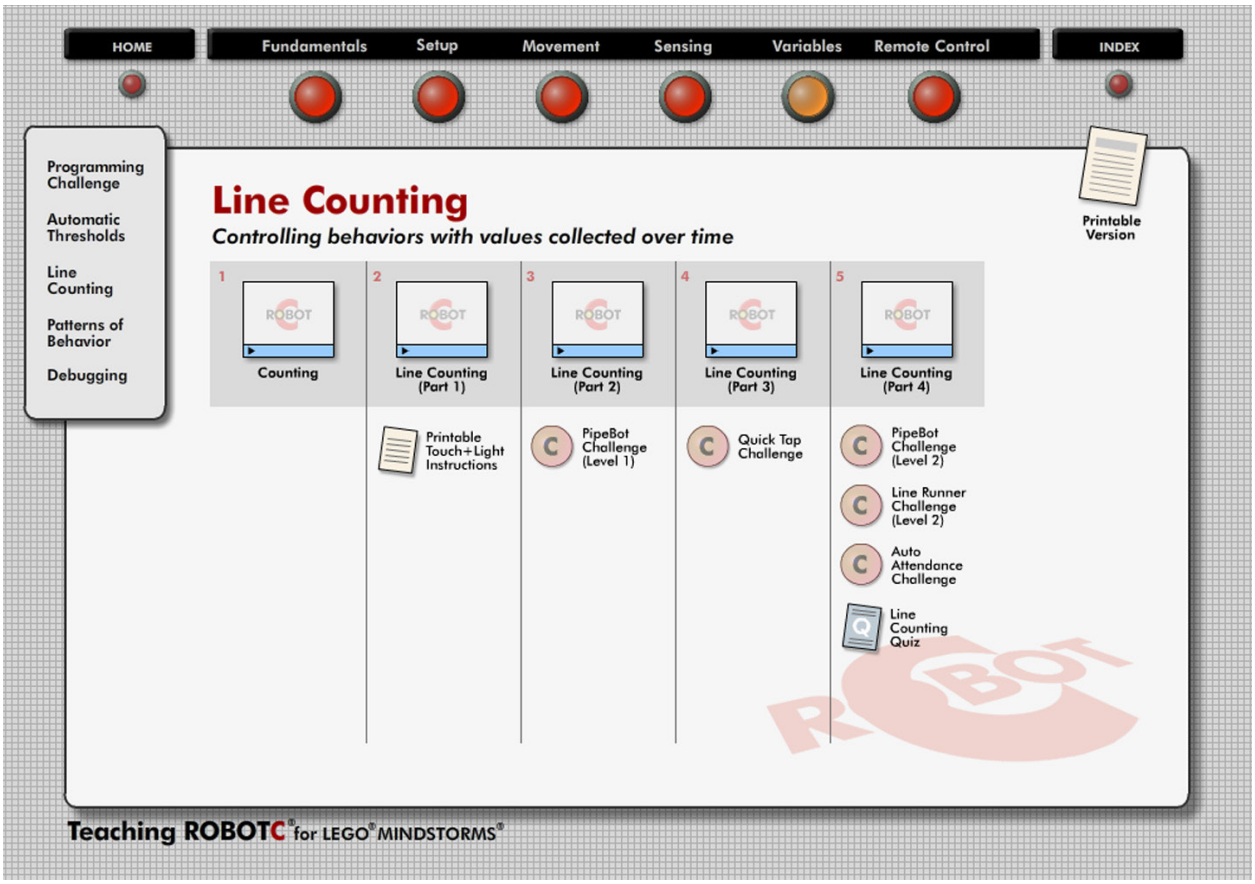






© 2009 Carnegie Mellon Robotics Academy

Programming/Variables and Functions/Counting



Sensing - Variables and Functions/Counting

The Line Counting Lesson Set introduces students to using variable to store values so that their robots can count. It appears that this would be a very simple task to implement, but there are multiple small problems to consider. Students will learn more about processors and implications of processing speed that they may not have thought about. Students will learn to setup a counter variable, when to count, how to count, and how to stop counting in a program.

The Counting Video - This video introduces students to reassigning values to variables allowing the robot to count.

Line Counting Part 1 Video - This video helps student to identify the problem as well begin to write their code.

Line Counting Part 2 Video - This video introduces students to using the “breakpoint” function built into ROBOTC’s debugger allowing students to see how fast the robot’s processor loops.

Line Counting Part 3 Video - Students are shown how to create a set of variables and turn them on and off helping them to count or not count at the correct spots in their program.

Line Counting Part 4 Video - The last video in the Lesson Set shows students how they can control when the robot stops.

Line Counting PDF Lesson Set - A printed version of everything that was found in the Line Counting video lessons.

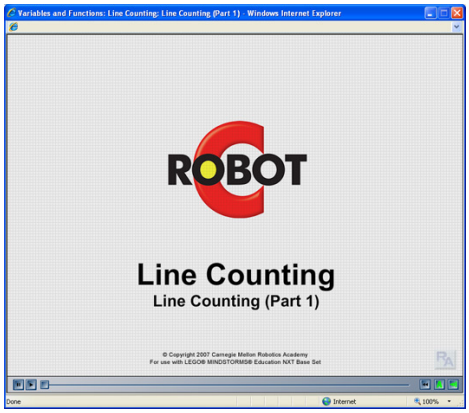
Programming Challenges - Students will begin to separate at this point. There are five programming challenges included in this Lesson Set. The programming challenges are more difficult at this point and not every student will complete every challenge. The challenges are: PipeBot Challenge (Level1), Quick Tap Challenge, The Pipe Bot Challenge (Level2), the Line Runner Challenge (Level 2), and the Auto Attendance Challenge.

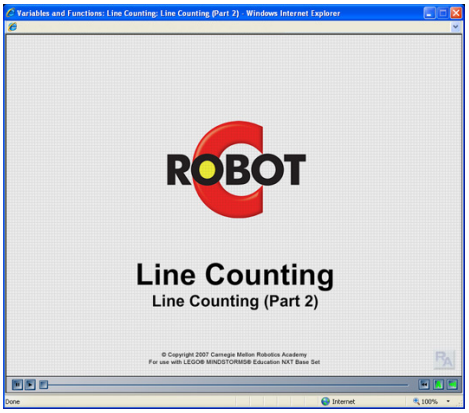
Line Counting Lesson Set Quiz - A quiz designed to check students understanding of these concepts.

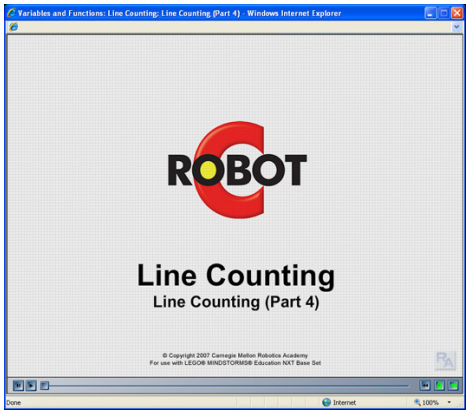
© 2009 Carnegie Mellon Robotics Academy

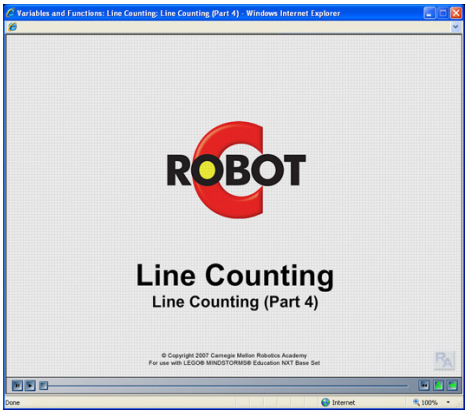
Programming/Variables and Functions/Counting

Sensing - Variables and Functions/Counting - Video Resources





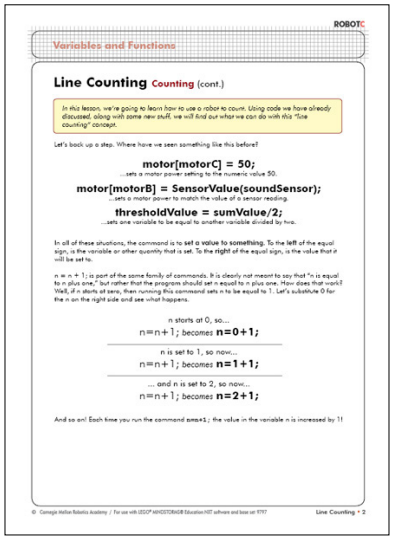




© 2009 Carnegie Mellon Robotics Academy

Programming/Variables and Functions/Counting

Sensing - Variables and Functions/Counting - Printed Resources



ROBOTC

Variables and Functions

Line Counting Counting (cont.)

In this lesson, we're going to learn how to use a robot to count. Using code we have already discussed, along with some new stuff, we will find out what we can do with this "line counting" concept.

Let's back up a step. Where have we seen something like this before?

motor[motorC] = 50;
...sets a motor power setting to the numeric value 50.

motor[motorB] = SensorValue(soundSensor);
...sets a motor power to match the value of a sensor reading.

thresholdValue = sumValue/2;
...sets one variable to be equal to another variable divided by two.

In all of these situations, the command is to **set a value to something.** To the **left** of the equal sign, is the variable or other quantity that is set. To the **right** of the equal sign, is the value that it will be set to.

n = n + 1; is part of the same family of commands. It is clearly not meant to say that "n is equal to n plus one," but rather that the program should set n equal to n plus one. How does that work? Well, if n starts at zero, then running this command sets n to be equal to 1. Let's substitute 0 for the n on the right side and see what happens.

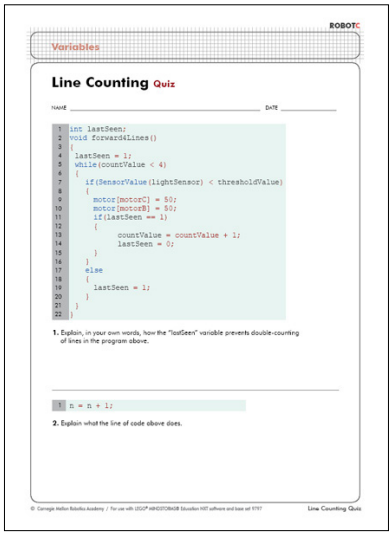
n starts at 0, so...
n=n+1; becomes **n=0+1;**

n is set to 1, so now...
n=n+1; becomes **n=1+1;**

... and n is set to 2, so now...
n=n+1; becomes **n=2+1;**

And so on! Each time you run the command n=n+1; the value in the variable n is increased by 1!

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Line Counting • 2



ROBOTC

Variables

Line Counting Quiz

NAME _____ DATE _____

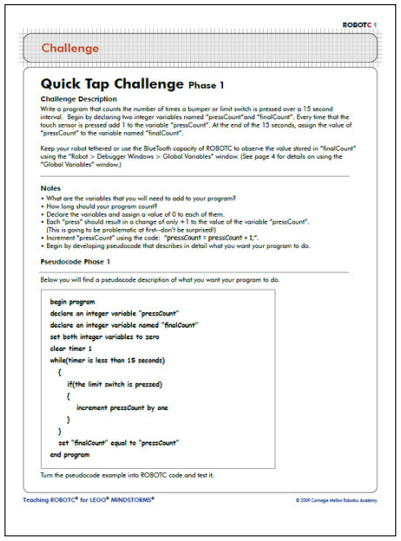
```
1  int lastSeen;
2  void forward4Lines()
3  {
4    lastSeen = 1;
5    while(countValue < 4)
6    {
7      if(SensorValue(lightSensor) < thresholdValue)
8      {
9        motor[motorC] = 50;
10       motor[motorB] = 50;
11       if(lastSeen == 1)
12       {
13           countValue = countValue + 1;
14           lastSeen = 0;
15       }
16     }
17     else
18     {
19       lastSeen = 1;
20     }
21   }
22 }
```

1. Explain, in your own words, how the "lastSeen" variable prevents double-counting of lines in the program above.

```
1  n = n + 1;
```

2. Explain what the line of code above does.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Line Counting Quiz



ROBOTC 1

Challenge

Quick Tap Challenge Phase 1

Challenge Description

Write a program that counts the number of times a bumper or limit switch is pressed over a 15 second interval. Begin by declaring two integer variables named "pressCount"and "finalCount". Every time that the touch sensor is pressed add 1 to the variable "pressCount". At the end of the 15 seconds, assign the value of "pressCount" to the variable named "finalCount".

Keep your robot tethered or use the BlueTooth capacity of ROBOTC to observe the value stored in "finalCount" using the "Robot > Debugger Windows > Global Variables" window. (See page 4 for details on using the "Global Variables" window.)

Notes

- What are the variables that you will need to add to your program?
- How long should your program count?
- Declare the variables and assign a value of 0 to each of them.
- Each "press" should result in a change of only +1 to the value of the variable "pressCount". (This is going to be problematic at first–don't be surprised!)
- Increment "pressCount" using the code: "pressCount = pressCount + 1;".
- Begin by developing pseudocode that describes in detail what you want your program to do.

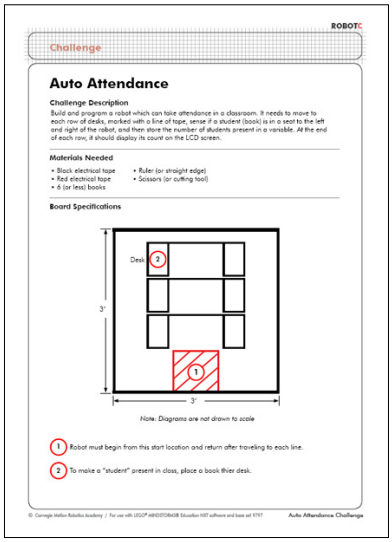
Pseudocode Phase 1

Below you will find a pseudocode description of what you want your program to do.

```
begin program
declare an integer variable "pressCount"
declare an integer variable named "finalCount"
set both integer variables to zero
clear timer 1
while(timer is less than 15 seconds)
  {
    if(the limit switch is pressed)
    {
      increment pressCount by one
    }
  }
  set "finalCount" equal to "pressCount"
end program
```

Turn the pseudocode example into ROBOTC code and test it.

Teaching ROBOTC® for LEGO® MINDSTORMS® — © 2009 Carnegie Mellon Robotics Academy



ROBOTC

Challenge

Auto Attendance

Challenge Description

Build and program a robot which can take attendance in a classroom. It needs to move to each row of desks, marked with a line of tape, sense if a student (book) is in a seat to the left and right of the robot, and then store the number of students present in a variable. At the end of each row, it should display its count on the LCD screen.

Materials Needed

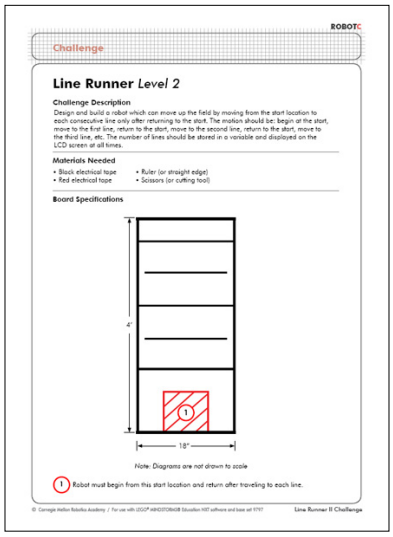
- Black electrical tape
- Red electrical tape
- 6 (or less) books
- Ruler (or straight edge)
- Scissors (or cutting tool)

Board Specifications

Note: Diagrams are not drawn to scale

1. Robot must begin from this start location and return after traveling to each line.
2. To make a "student" present in class, place a book thier desk.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Auto Attendance Challenge



ROBOTC

Challenge

Line Runner *Level 2*

Challenge Description

Design and build a robot which can move up the field by moving from the start location to each consecutive line only after returning to the start. The motion should be: begin at the start, move to the first line, return to the start, move to the second line, return to the start, move to the third line, etc. The number of lines should be stored in a variable and displayed on the LCD screen at all times.

Materials Needed

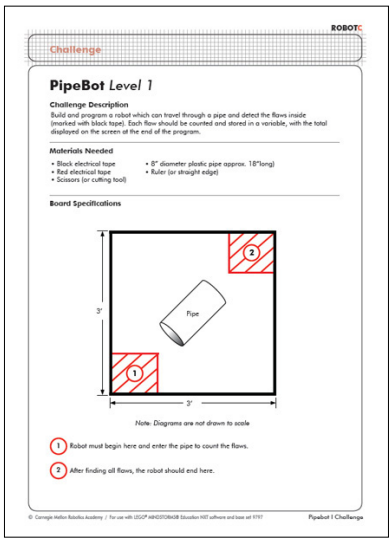
- Black electrical tape
- Red electrical tape
- Ruler (or straight edge)
- Scissors (or cutting tool)

Board Specifications

Note: Diagrams are not drawn to scale

1. Robot must begin from this start location and return after traveling to each line.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Line Runner II Challenge



ROBOTC

Challenge

PipeBot *Level 1*

Challenge Description

Build and program a robot which can travel through a pipe and detect the flaws inside (marked with black tape). Each flaw should be counted and stored in a variable, with the total displayed on the screen at the end of the program.

Materials Needed

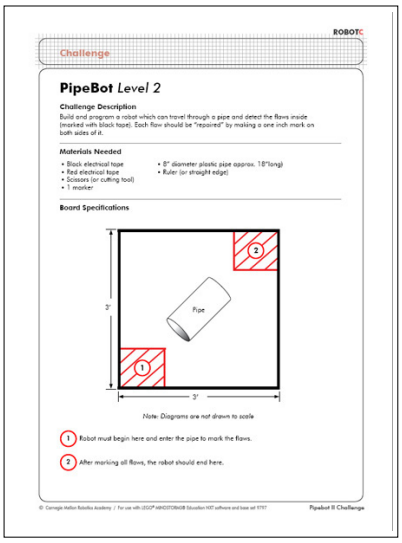
- Black electrical tape
- Red electrical tape
- Scissors (or cutting tool)
- 8" diameter plastic pipe approx. 18"long)
- Ruler (or straight edge)

Board Specifications

Note: Diagrams are not drawn to scale

1. Robot must begin here and enter the pipe to count the flaws.
2. After finding all flaws, the robot should end here.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Pipebot I Challenge



ROBOTC

Challenge

PipeBot *Level 2*

Challenge Description

Build and program a robot which can travel through a pipe and detect the flaws inside (marked with black tape). Each flaw should be "repaired" by making a one inch mark on both sides of it.

Materials Needed

- Black electrical tape
- Red electrical tape
- Scissors (or cutting tool)
- 1 marker
- 8" diameter plastic pipe approx. 18"long)
- Ruler (or straight edge)

Board Specifications

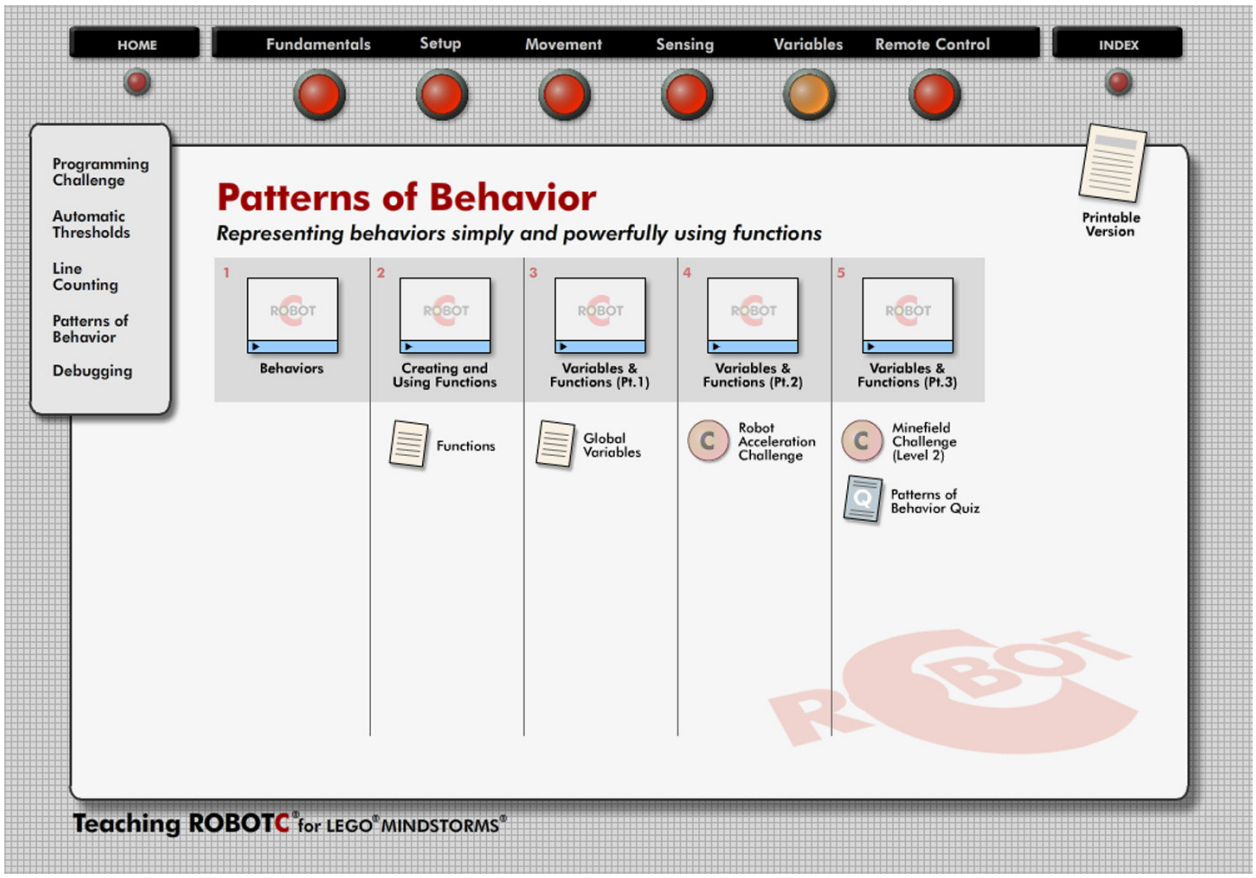
Note: Diagrams are not drawn to scale

1. Robot must begin here and enter the pipe to mark the flaws.
2. After marking all flaws, the robot should end here.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 — Pipebot II Challenge

© 2009 Carnegie Mellon Robotics Academy

Programming/Variables and Functions/Patterns of Behaviors



Sensing - Variables and Functions/Patterns of Behaviors

In the Patterns and Behaviors Lesson Set students learn to program using functions. Functions are very powerful organizers used by programmers that program robots using behavior based programming strategies.

Behaviors Video - Introductory video that explains the value of programming using functions.

Creating and Using Functions Video - In this video students learn the relationship between task main and a function. They learn to declare functions and use them in the task main section of their program.

Variables and Functions Part 1 Video - In this video students learn how to take code from other programs and turn them into functions and then they will take those functions and use them in an easier to read program. They also learn about the scope of variables and how to create global variables.

Variables and Functions Part 2 Video - In this video students continue to write the solution to the “warehouse” challenge using their newly found function tool.

Variables and Functions Part 3 Video - In this video students learn to pass parameters making their functions much more powerful.

Functions Reference Guide - This is a multipage handout that shows students how functions work.

Global Variables Reference Guide - This handout explains what a global variable is and how it can be used.

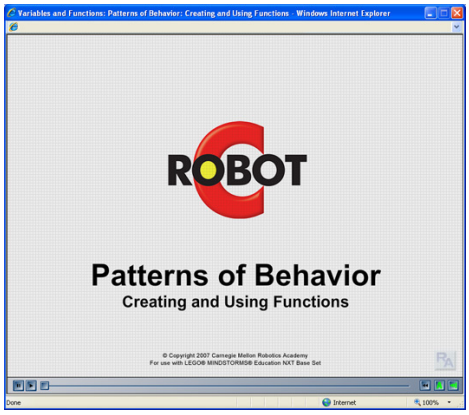
Programming Challenges - This Lesson Set include two advanced programming challenges: the Robot Acceleration Challenge, and the Minefield Challenge (Level2).

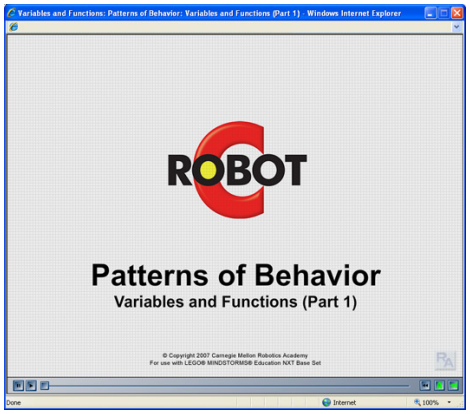
The Patterns of Behaviors Lesson Set Handout - This is a 40 page PDF that complements the videos in this Lesson Set.

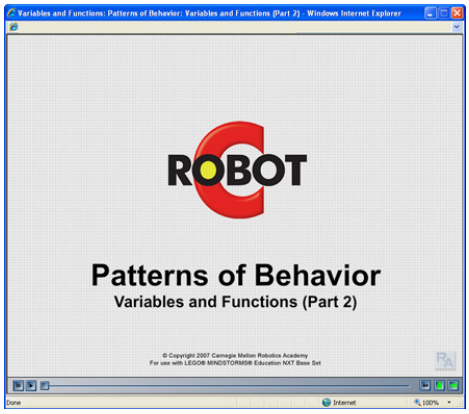
© 2009 Carnegie Mellon Robotics Academy

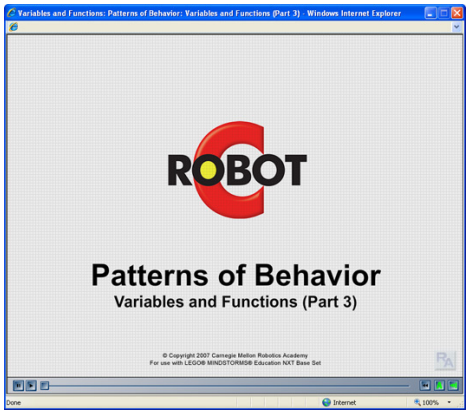
Programming/Variables and Functions/Patterns of Behaviors

Sensing - Variables and Functions/Patterns of Behaviors - Video Resources





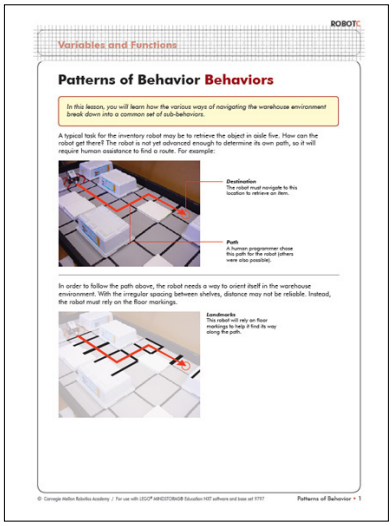




© 2009 Carnegie Mellon Robotics Academy

Programming/Variables and Functions/Patterns of Behaviors

Sensing - Variables and Functions/Patterns of Behaviors - Print Resources



ROBOTC

Variables and Functions

Patterns of Behavior Behaviors

In this lesson, you will learn how the various ways of navigating the warehouse environment break down into a common set of sub-behaviors.

A typical task for the inventory robot may be to retrieve the object in aisle five. How can the robot get there? The robot is not yet advanced enough to determine its own path, so it will require human assistance to find a route. For example:

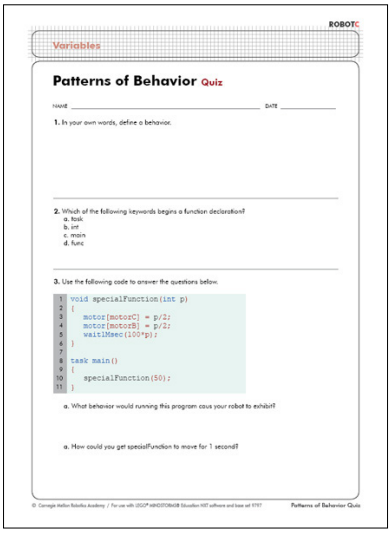
Destination
The robot must navigate to this location to retrieve an item.

Path
A human programmer chose this path for the robot (others were also possible).

In order to follow the path above, the robot needs a way to orient itself in the warehouse environment. With the irregular spacing between shelves, distance may not be reliable. Instead, the robot must rely on the floor markings.

Landmarks
This robot will rely on floor markings to help it find its way along the path.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Patterns of Behavior • 1



ROBOTC

Variables

Patterns of Behavior Quiz

NAME _____ DATE _____

1. In your own words, define a behavior.

2. Which of the following keywords begins a function declaration?
a. task
b. int
c. main
d. func

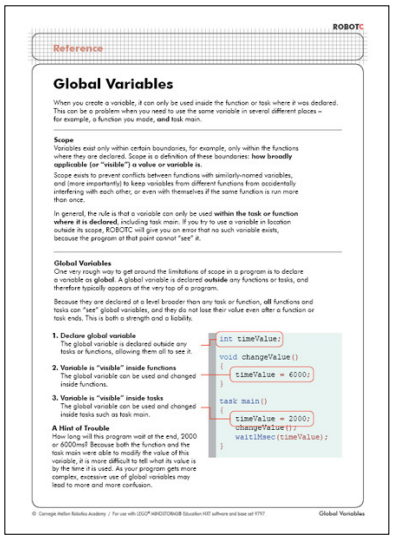
3. Use the following code to answer the questions below.

```
1  void specialFunction(int p)
2  {
3    motor[motorC] = p/2;
4    motor[motorB] = p/2;
5    wait1Msec(100*p);
6  }
7
8  task main()
9  {
10   specialFunction(50);
11 }
```

a. What behavior would running this program cause your robot to exhibit?

a. How could you get specialFunction to move for 1 second?

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Patterns of Behavior Quiz



ROBOTC

Reference

Global Variables

When you create a variable, it can only be used inside the function or task where it was declared. This can be a problem when you need to use the same variable in several different places – for example, a function you made, **and** task main.

Scope
Variables exist only within certain boundaries, for example, only within the functions where they are declared. Scope is a definition of these boundaries: **how broadly applicable (or "visible") a value or variable is.**

Scope exists to prevent conflicts between functions with similarly-named variables, and (more importantly) to keep variables from different functions from accidentally interfering with each other; or even with themselves if the same function is run more than once.

In general, the rule is that a variable can only be used **within the task or function where it is declared**, including task main. If you try to use a variable in location outside its scope, ROBOTC will give you an error that no such variable exists, because the program at that point cannot "see" it.

Global Variables
One very rough way to get around the limitations of scope in a program is to declare a variable as **global**. A global variable is declared **outside** any functions or tasks, and therefore typically appears at the very top of a program.

Because they are declared at a level broader than any task or function, **all** functions and tasks can "see" global variables, and they do not lose their value even after a function or task ends. This is both a strength and a liability.

1. Declare global variable
The global variable is declared outside any tasks or functions, allowing them all to see it.

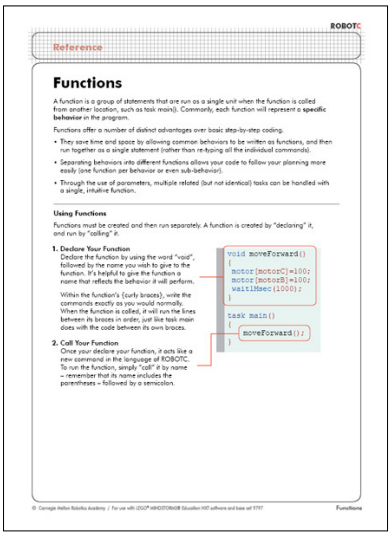
2. Variable is "visible" inside functions
The global variable can be used and changed inside functions.

3. Variable is "visible" inside tasks
The global variable can be used and changed inside tasks such as task main.

```
int timeValue;

void changeValue()
{
  timeValue = 6000;
}

task main()
{
  timeValue = 2000;
  changeValue();
  waitMsec(timeValue);
}
```

A Hint of Trouble
How long will this program wait at the end, 2000 or 6000ms? Because both the function and the task main were able to modify the value of this variable, it is more difficult to tell what its value is by the time it is used. As your program gets more complex, excessive use of global variables may lead to more and more confusion.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Global Variables



ROBOTC

Reference

Functions

A function is a group of statements that are run as a single unit when the function is called from another location, such as task main(). Commonly, each function will represent a **specific behavior** in the program.

Functions offer a number of distinct advantages over basic step-by-step coding.

- They save time and space by allowing common behaviors to be written as functions, and then run together as a single statement (rather than re-typing all the individual commands).
- Separating behaviors into different functions allows your code to follow your planning more easily (one function per behavior or even sub-behavior).
- Through the use of parameters, multiple related (but not identical) tasks can be handled with a single, intuitive function.

Using Functions
Functions must be created and then run separately. A function is created by "declaring" it, and run by "calling" it.

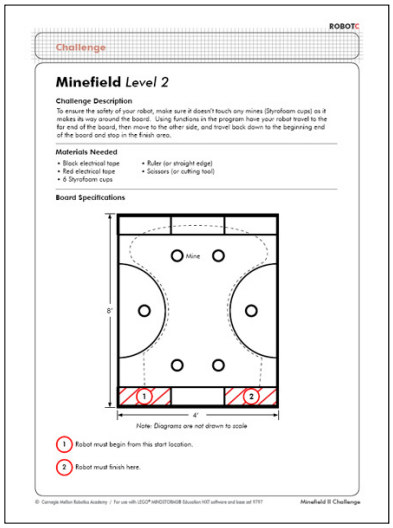
1. Declare Your Function
Declare the function by using the word "void", followed by the name you wish to give to the function. It's helpful to give the function a name that reflects the behavior it will perform.

Within the function's {curly braces}, write the commands exactly as you would normally. When the function is called, it will run the lines between its braces in order, just like task main does with the code between its own braces.

2. Call Your Function
Once your declare your function, it acts like a new command in the language of ROBOTC. To run the function, simply "call" it by name – remember that its name includes the parentheses – followed by a semicolon.

```
void moveForward()
{
  motor[motorC]=100;
  motor[motorB]=100;
  waitMsec(1000);
}

task main()
{
  moveForward();
}
```

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Functions



ROBOTC

Challenge

Minefield *Level 2*

Challenge Description
To ensure the safety of your robot, make sure it doesn't touch any mines (Styrofoam cups) as it makes its way around the board. Using functions in the program have your robot travel to the far end of the board, then move to the other side, and travel back down to the beginning end of the board and stop in the finish area.

Materials Needed
- Black electrical tape
- Red electrical tape
- 6 Styrofoam cups
- Ruler (or straight edge)
- Scissors (or cutting tool)

Board Specifications

Mine

8'

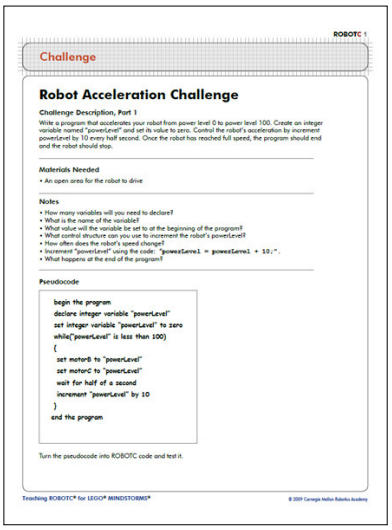
4'

Note: Diagrams are not drawn to scale

1 Robot must begin from this start location.

2 Robot must finish here.

© Carnegie Mellon Robotics Academy / For use with LEGO® MINDSTORMS® Education NXT software and base set 9797 Minefield II Challenge



ROBOTC

Challenge

Robot Acceleration Challenge

Challenge Description, Part 1
Write a program that accelerates your robot from power level 0 to power level 100. Create an integer variable named "powerLevel" and set its value to zero. Control the robot's acceleration by increment powerLevel by 10 every half second. Once the robot has reached full speed, the program should end and the robot should stop.

Materials Needed
- An open area for the robot to drive

Notes
- How many variables will you need to declare?
- What is the name of the variable?
- What value will the variable be set to at the beginning of the program?
- What control structure can you use to increment the robot's powerLevel?
- How often does the robot's speed change?
- Increment "powerLevel" using the code: `"powerLevel = powerLevel + 10;"`.
- What happens at the end of the program?

Pseudocode

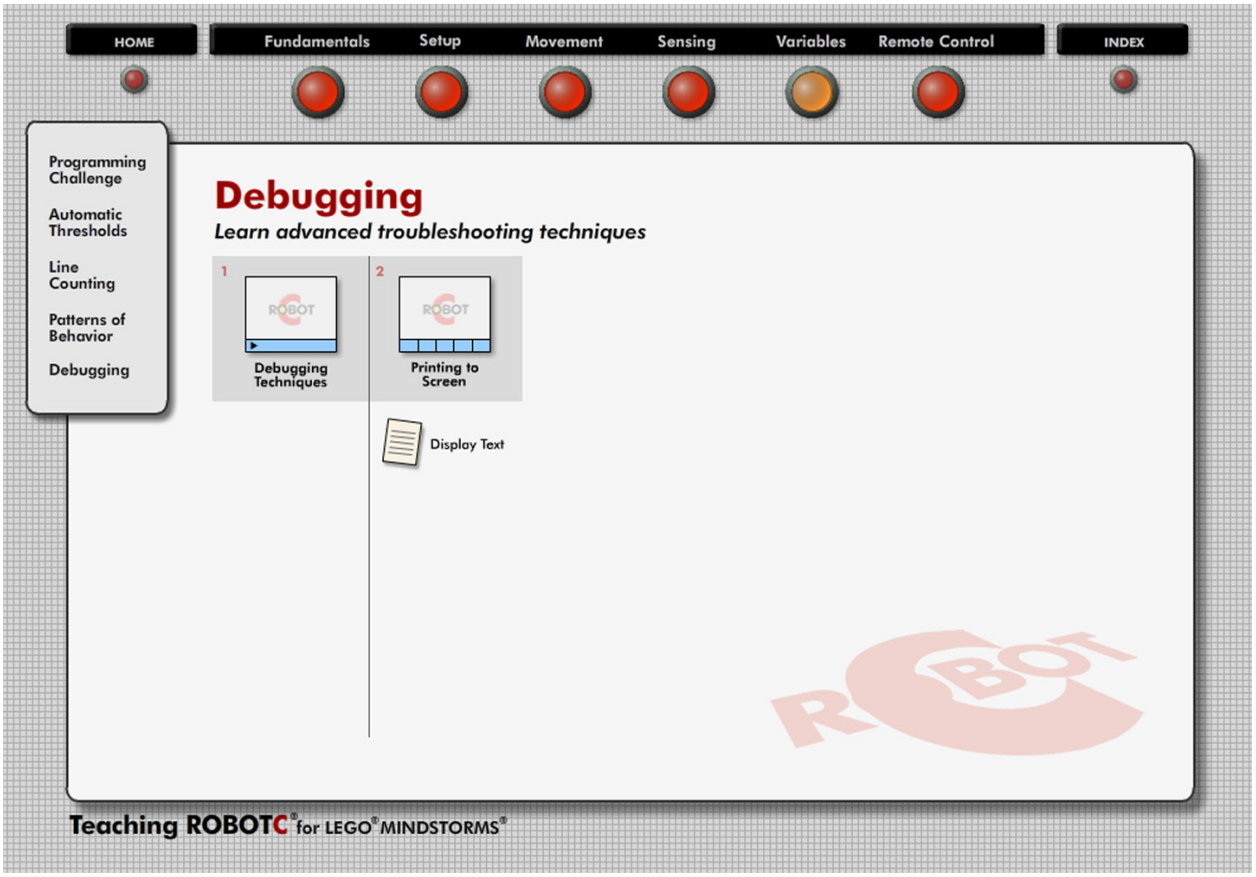
```
begin the program
declare integer variable "powerLevel"
set integer variable "powerLevel" to zero
while("powerLevel" is less than 100)
{
 set motorB to "powerLevel"
 set motorC to "powerLevel"
 wait for half of a second
 increment "powerLevel" by 10
}
end the program
```

Turn the pseudocode into ROBOTC code and test it.

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

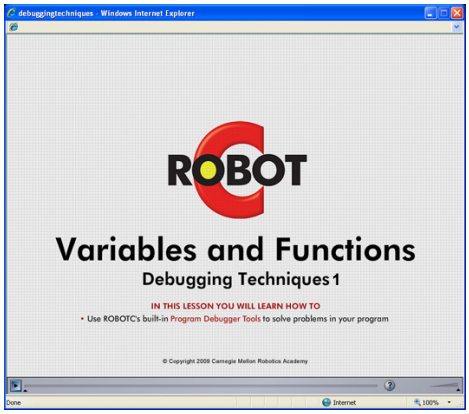
© 2009 Carnegie Mellon Robotics Academy

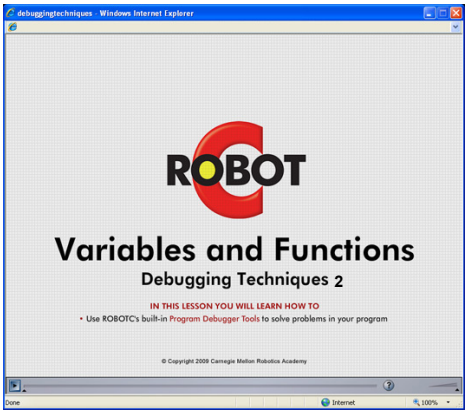
Programming/Variables and Functions/Debugging



Sensing - Variables and Functions/Debugging

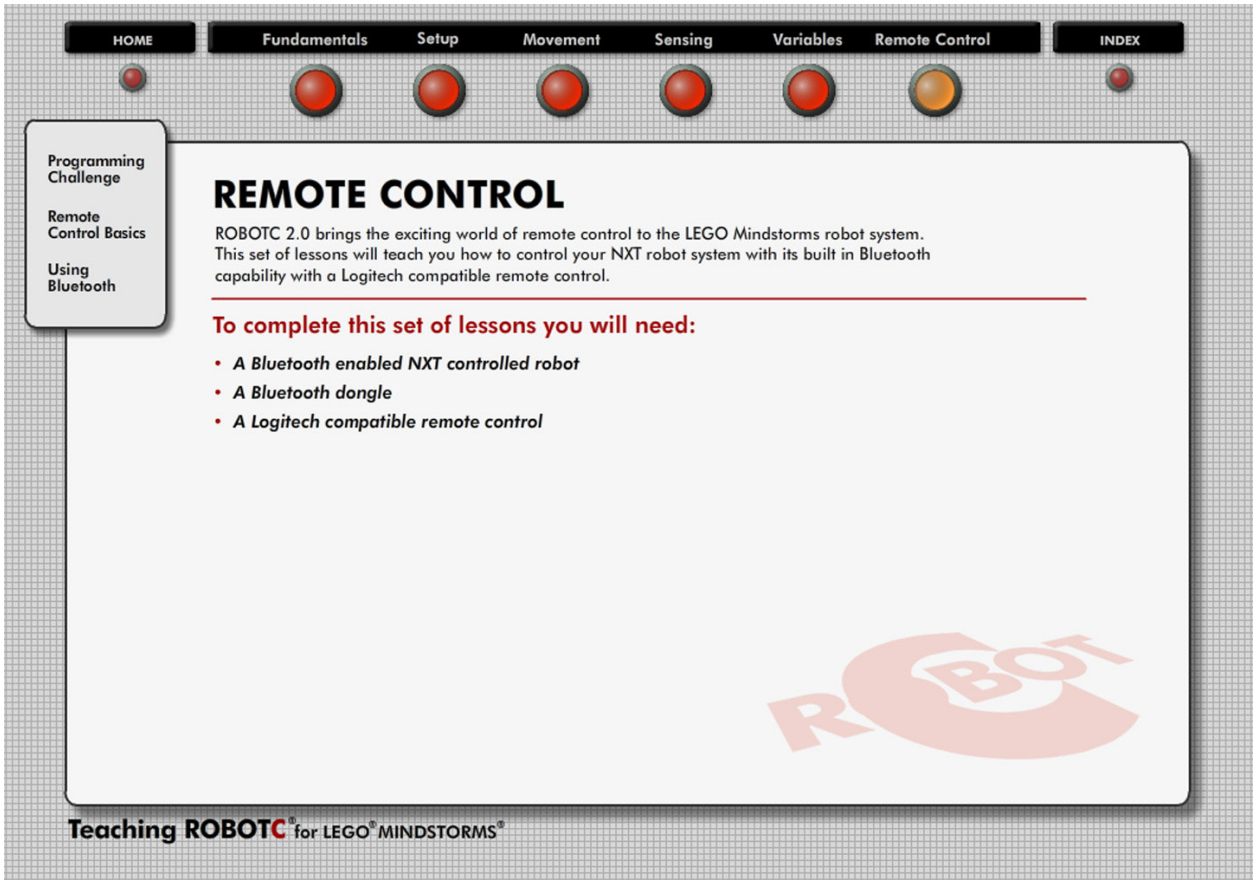
The Debugging Lesson Set consists of a set of videos designed to highlight ROBOTC's debugging screen. The first video, Debugging Techniques has the student open up a sample program that shows them how to open up the expert mode to use the higher level debugging tools available through ROBOTC. Students will learn how to use the step into, step over, step out, breakpoint, and clear all features found on the debug start menu. The second video set, Printing to Screen, highlights the advantage of using the on screen NXT display so that you can see values as you troubleshoot your code.





© 2009 Carnegie Mellon Robotics Academy

Programming/Remote Control



Programming/Remote Control

The Remote Control Unit teaches students how Bluetooth works, how to troubleshoot Bluetooth, and how to program their remote control unit. ROBOTC allows two Logitech remote control units to control one robot. All of the lessons in this curriculum are designed for one remote control unit. For more information on controlling one robot with two remote control units refer to the built in help in ROBOTC.

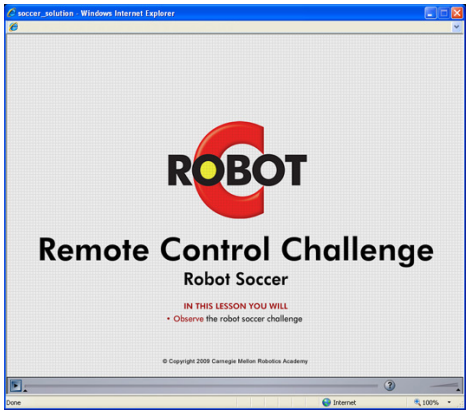
© 2009 Carnegie Mellon Robotics Academy

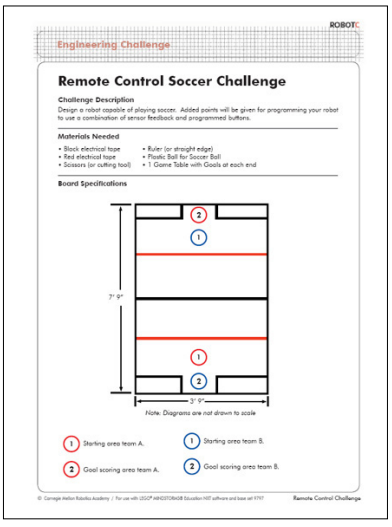
Programming/Remote Control/Programming Challenge

Programming/Remote Control

The Remote Control Robot Soccer Programming Challenge provides a fun engineering/programming challenge that students of all levels enjoy. In the Remote Control programming unit, students will learn how to program the joysticks and buttons on their Logitech remote control units to produce desired behaviors like “kick the ball” or “drive straight”. They already know how to program using the LEGO sensors, but there are a whole array of other third party sensors available that make the robot soccer challenge a great engineering/programming challenge. Two sensors that come to mind quickly are the compass sensor and the IR sensor. The soccer challenge is a very good challenge at all levels middle school through adult hobbyist.

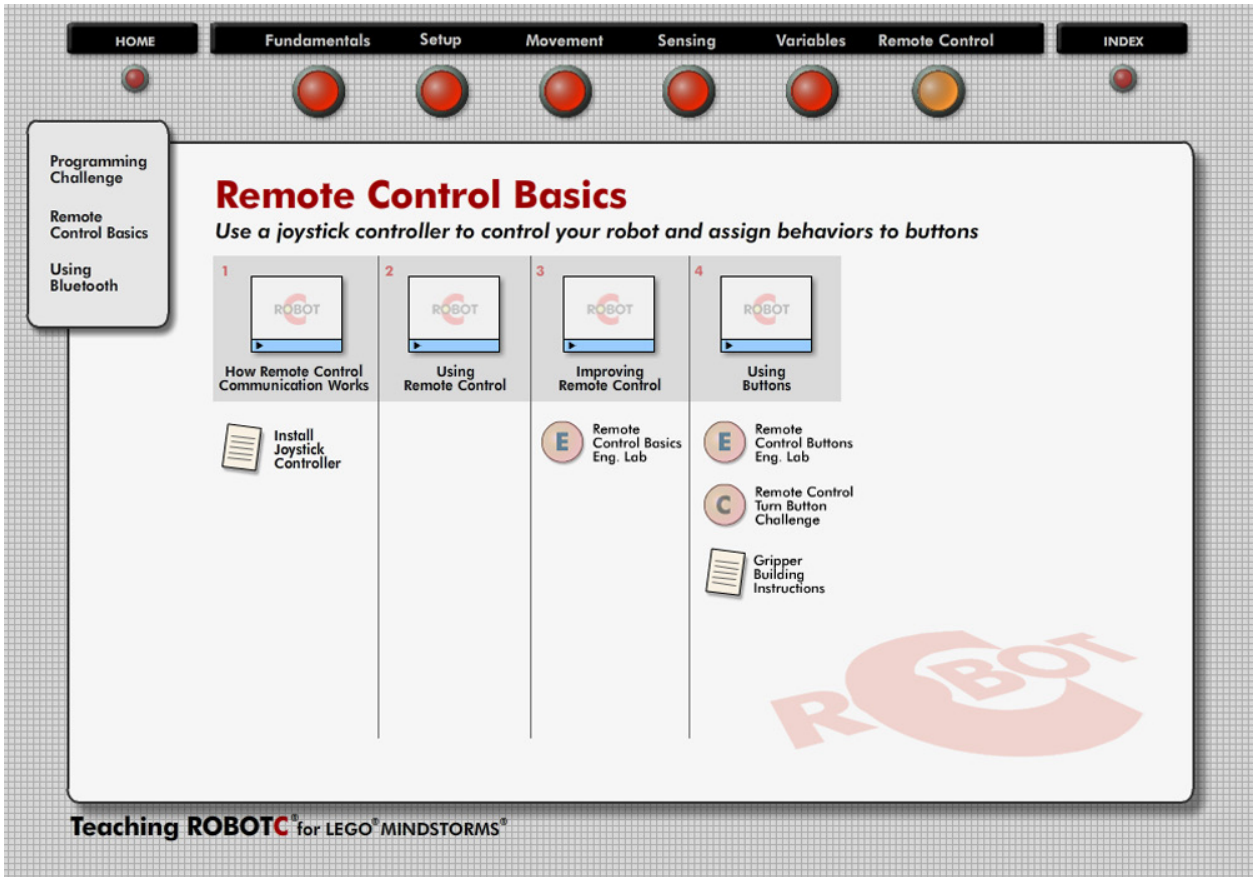
The following resources are available to preview the challenge:





© 2009 Carnegie Mellon Robotics Academy

Programming/Remote Control/Remote Control Basics



Remote Control Basics

The Remote Control Basics Lesson Set teaches students everything that they need to know to combine autonomous and remote control programming. This is a very powerful addition to the LEGO's already powerful NXT hardware. This Lesson Set contains two Remote Control Engineering Labs where students will be guided step-by-step toward a successful implementation of a combination of remote control and autonomous programming. The following resources are available to support this Lesson Set:

How Remote Control Communications Works Video - This video describes the communications between the Logitech Remote Control, the computer, and the NXT based robot.

Using Remote Control Video - In this lesson, students learn about include files, the new functions that are made available via the include file, and how the new functions map to the Logitech remote control.

Improving Remote Control Video - This lesson teaches students how to improve the joystick control via variable math and creating a dead zone on their remote control.

Using Buttons Video - This lesson teaches students how to program the buttons on the Logitech controller.

Install the Joystick Controller Reference - This is a reference PDF that helps to install and troubleshoot the Logitech remote controller installation.

Remote Control Basics Engineering Lab - This lab provides step-by-step instructions designed to help a student to learn how to program the Logitech remote controller joysticks to control the NXT.

Remote Control Buttons Engineering Lab - This lab provides step-by-step instructions on how to write a program that enables the Logitech remote controller buttons to control the NXT.

Gripper Building Instructions - A set of instruction for students to use to build a LEGO gripper.

© 2009 Carnegie Mellon Robotics Academy

Programming/Remote Control/Remote Control Basics

Remote Control Basics Resources

howworks - Windows Internet Explorer

ROBOT

Remote Control

How Remote Control Works

IN THIS LESSON YOU WILL LEARN

- How remote control works.

© Copyright 2008 Carnegie Mellon Robotics Academy

usingrc - Windows Internet Explorer

ROBOT

Remote Control

Using Remote Control

IN THIS LESSON YOU WILL LEARN

- How to use the joystick controller to send motor commands to the NXT .

© Copyright 2008 Carnegie Mellon Robotics Academy

ROBOTC 1

Engineering Lab

Remote Control Basics

In this lesson you will:

1. Learn how the remote control, computer, and NXT communicate.
2. Open a sample program named "NXT 2 Joystick Drive", drive your robot using that code, and then optimize the code.
3. Be able to:
   - Describe how NXT remote communications works.
   - Describe what an "include file" is.
   - Describe what the "getJoystickSettings" function does.
   - Describe what the variable "joystick.joy1_y1" is and how what makes it different from "joystick.joy1_y2".
   - Optimize your program for better joystick control.

How are the communication signals sent?

Until now, you have written code that allows your robot to operate autonomously. There are situations where a programmer cannot plan every possible behavior using autonomous programming; for these situations it would be nice to be able to use a combination of autonomous and remote control. ROBOTC makes remote control very easy.

How does remote control work with the NXT? There are four elements that work together: the human driver, the joystick controller, the computer with ROBOTC, and the NXT. When the joysticks axes are pushed in one direction or another, a signal is created based on the direction the joystick is pushed. The signal returns a value that ranges from +127 to -127. Once the computer gets the signal, it is sent to the NXT by ROBOTC, where the function "getJoystickSettings" receives that data and places it into a variable that can be used in our program. (i.e. to power the motors) The computer sends the joystick data to the NXT over the Bluetooth or USB link.

1. When the joysticks axes are pushed in one direction or another, a signal is created based on the location of the joystick; the signal ranges from +127 to -127.
2. The signal is sent to the computer, where ROBOTC converts the signal information into information the NXT can understand.
3. ROBOTC then sends the data to the NXT over Bluetooth or USB so the NXT can process and use the data to control its motors.

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

improvingrc - Windows Internet Explorer

ROBOT

Remote Control

Improving Remote Control

IN THIS LESSON YOU WILL LEARN

- How to create a dead-zone in order to improve the joystick controls.

© Copyright 2008 Carnegie Mellon Robotics Academy

usingbuttons - Windows Internet Explorer

ROBOT

Remote Control

Using Buttons

IN THIS LESSON YOU WILL LEARN

- How to program the joystick buttons in order to open and close the gripper.

© Copyright 2008 Carnegie Mellon Robotics Academy

ROBOTC 1

Engineering Lab

Remote Control Buttons

In this lesson you will:

1. Learn how to program the buttons on your remote controller.
2. Review the "JoystickDriver" include file.
3. Review the function "getJoystickSettings(joystick)".
4. Identify the names and locations of all buttons on the joystick.

Remote Control Button Overview

The joystick remote controller is a very powerful tool that a programmer can use to use to achieve finite control of their robot. Each button can be programmed to control a specific behavior, for example - goStraight, rightTurn, leftTurn, openGripper, closeGripper - allowing limitless options.

Button 9 Button 10

Front View
joystick.joy1

The Joystick TopHat - The joystick "TopHat" is usable in your program and accessed with the reserve word "joystick.joy1_TopHat". The TopHat returns values from -1 to 7. The -1 value signifies that nothing is pressed.

Top View
Buttons 5, 6, 7, & 8

Joysticks: The joystick controller is named joystick.joy1. There are two joystick axes on the joystick.joy1 controller. They are the round knobs that are labeled x1/y1 and x2/y2 on the picture on the left. To access the "y-axis" of joy_1 the command would be "joystick.joy1_y1".

The joystick axis names are:
joystick.joy1_y1
joystick.joy1_x1
joystick.joy1_y2
joystick.joy1_x2

Note: ROBOTC has the capability of working with two joystick controllers at a time. There is also a joystick.joy2 remote controller that can be programmed with ROBOTC.

Buttons: There are 10 programmable buttons on the joystick controller. They are found in three locations.

Buttons 1-4 are located on the front right side of the joystick controller.

Buttons 5-8 are located on the top of the joystick controller.

Buttons 9 and 10 are above the joysticks.

Joystick buttons values are accessed: joy1Btn(1) through joy1Btn(10)

Note: There is a set of joy2Btn(#) that map to the second joystick controller.

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

ROBOTC 1

Remote Control Basics

Logitech Joystick Controller Installation

Description
The USB Logitech Controller allows you to remotely control the movements of your NXT.
Use this guide to ensure that your hardware is properly installed on your computer.

Materials Needed

- 1 Logitech USB Controller (available at http://store.mbomatter.com/)
- 1 PC (Windows XP SP2+ or Windows Vista recommended)
- 1 Available USB Port

Logitech USB Controller Installation
Plug the Logitech USB Controller into an available USB port.

Plug in the Logitech Controller
Plug the Logitech controller into an open USB port on your computer.

Found New Hardware
Your new hardware is installed and ready to use.

Windows Installation
Windows will automatically install the Logitech controller. This process may take a few minutes.

Notes:

- Use ONLY the Logitech USB controllers! Other USB controllers are not supported and may not function properly.

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

Building Instructions

Gripper

© Carnegie Mellon Robotics Academy

ROBOTC

Challenge

Remote Control - Turn Button Challenge

Challenge Description

Write a program that uses a combination of remote control and programmed buttons on the joystick to complete the course below. Incorporate the following robot functions into your program, having them activate with buttons:
rightTurn - leftTurn - straightForward - reverse
Use joystick control for other driving movements

Materials Needed

- Black electrical tape
- Scissors (or cutting tool)

Board Specifications

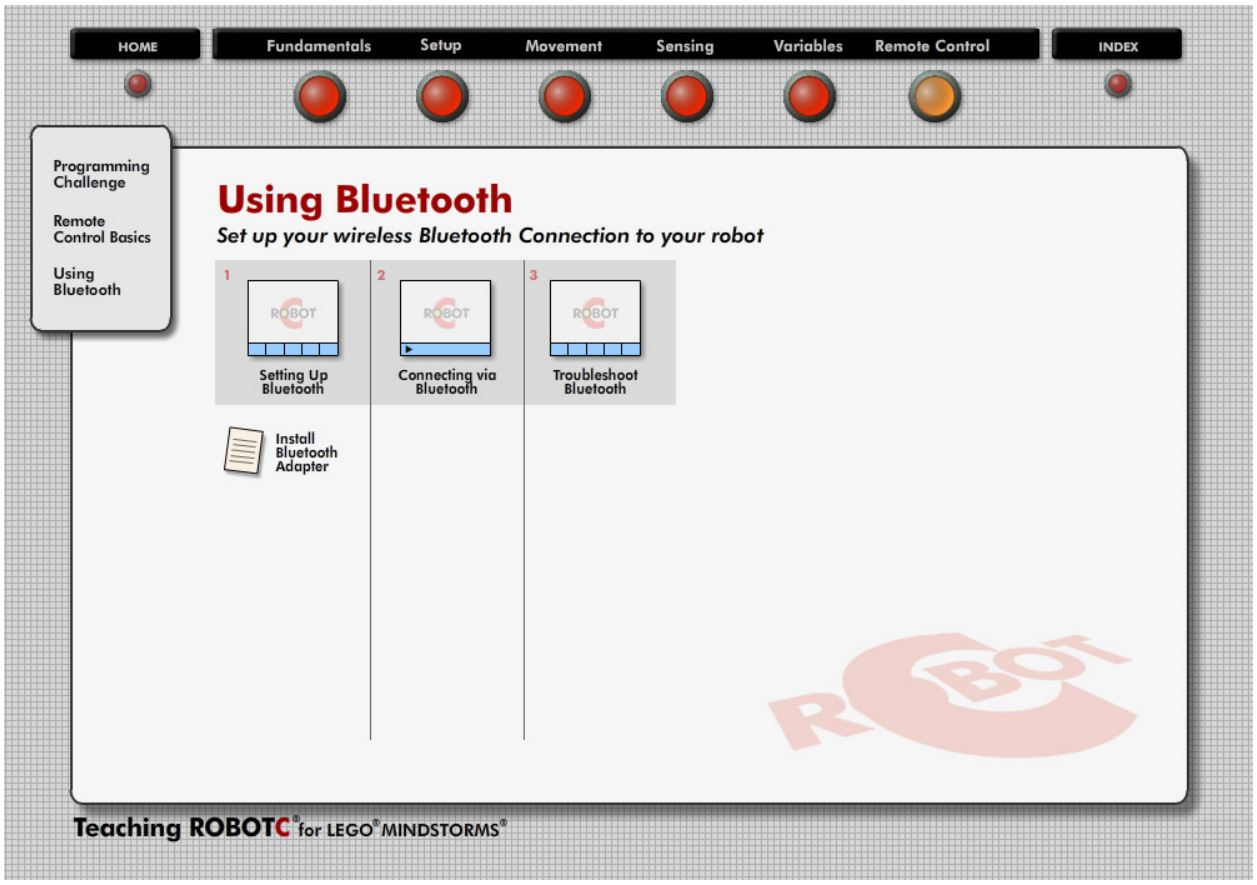
Finish
8'
Start
4'

Note: Diagrams are not drawn to scale

Teaching ROBOTC® for LEGO® MINDSTORMS® © 2009 Carnegie Mellon Robotics Academy

© 2009 Carnegie Mellon Robotics Academy

Programming/Remote Control/Using Bluetooth



Using Bluetooth

The Using Bluetooth Lesson Set is designed to help a first time user connect their computer to their NXT. There are four resources available in this section that have multiple components designed to ensure success.

The Setting Up Bluetooth Video Set - This Lesson Set focuses on answering the following questions: What is Bluetooth? How do I rename my NXT so that it is easy for Bluetooth to recognize? and How do I ensure that my NXT is visible to my computer?

The Connecting Via Bluetooth Video - This video demonstrates step-by-step how to connect via Bluetooth using ROBOTC.

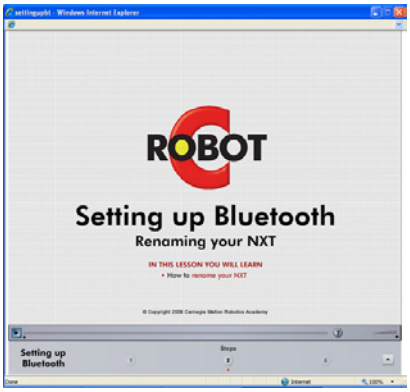
The Troubleshooting Bluetooth Video Set - This troubleshooting guide consists of six things to consider if your Bluetooth connection isn't working.

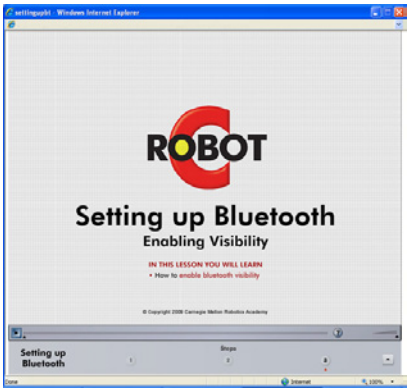
Installing your Bluetooth Adaptor Reference Guide - A one page handout that describes the Bluetooth adaptor.

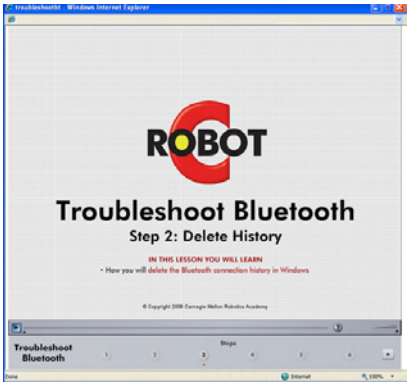
© 2009 Carnegie Mellon Robotics Academy

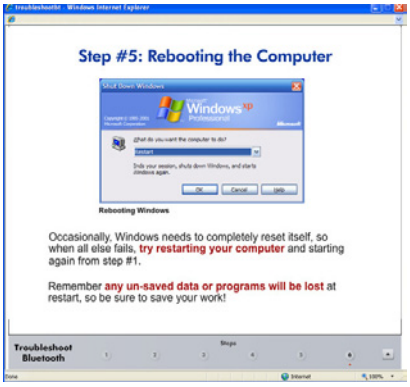
Programming/Remote Control/Using Bluetooth

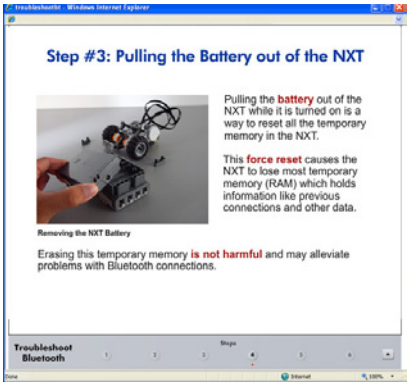
Using Bluetooth Resources











© 2009 Carnegie Mellon Robotics Academy

How Robotics Aligns with Standards

This section describes how robotics as a content area aligns with National Science, Mathematics, Technology, and Communications Standards. Below you will see the format that we use to show how robotics education aligns to national science, technology, engineering, and mathematics (STEM) standards.

Standard	Robotics Link
On the left is a description of the standard or particular point of the standard that is addressed through robotics.	On the right is a description of how robotics in general and this curriculum in particular addresses this standard.

Science Standards Addressed

From the National Science Education Standards (NSES)

Systems, Order, and Organization

The natural and designed world is complex; it is too large and complicated to investigate and comprehend all at once.

A system is an organized group of related objects or components that form a whole.

The goal of this standard is to think and analyze in terms of systems.

Science assumes that the behavior of the universe is not capricious, that nature is the same everywhere, and that it is understandable and predictable.

Prediction is the use of knowledge to identify and explain observation, or changes, in advance. The use of mathematics allows for greater or lesser certainty of predictions.

Order is the behavior of units of matter, objects, organisms or events in the universe – can be described mathematically.

Types and levels of organization provide useful ways of thinking about the world

Robots are excellent examples of systems, with many heterogeneous components interacting in organized, methodical ways to achieve results as a whole that they could not have achieved separately.

Examples include:
- Navigation systems (e.g. sensor tells the robot where it is, programmable controller tells the robot how to interpret this information, motors move in order to achieve the desired result)
- Sensing systems (electrical, mechanical, and programming elements of a sensor)
- Power & transmission systems (motor, axle, gear, wheel)
- Manipulator systems
- Lifting systems, vision systems, etc.

Each system can be broken down into subsystems.

Robotics technology is built upon a series of behaviors that can be measured mathematically and are understandable and predictable.

There are many examples that are easy for students to manipulate and understand:
- Gears and mechanical advantage
- Sensors and electronic control
- Wheel diameter and its effect on distance traveled
- Rotation sensor readings and robot path planning

© 2009 Carnegie Mellon Robotics Academy

Science Standards continued

From the National Science Education Standards (NSES)

Evidence, Models and Explanation

Evidence consists of observations and data on which to base scientific explanations. Using evidence to understand interactions allows individuals to predict changes in natural and designed systems.

Models are tentative schemes or structures that correspond to real objects, events, or classes of events that have explanatory power. Models help scientists and engineers understand how things work. Models take many forms, including physical objects, plans, mental constructs, mathematical equations and computer simulations.

Scientific explanations incorporate existing scientific knowledge and new evidence into logical statements. Terms like "hypothesis," "model," "law," "theory," and "paradigm" are used to describe various scientific explanations.

The investigations included in this curriculum allow students to collect evidence to investigate scientific principles. Robots physically demonstrate many scientific concepts to make them more clear and understandable.

Examples include:

- Electronics and basic circuitry, which can be demonstrated using touch sensors and the NXT power supply
- Gear trains, which demonstrate the ability to mathematically predict mechanical advantage and speed.
- Light sensors, which can detect infrared as well as visible light

Constancy, Change and Measurement

Although most things are in the process of becoming different – changing – some properties of objects and processes are characterized by constancy; the speed of light, the charge of an electron, the total mass plus energy of the universe.

Energy can be transmitted and matter can be changed. Nevertheless, when measured, the sum of energy and matter in the system, and, by extension, the universe, remains the same.

Mathematics is essential for accurately measuring change.

Different systems of measurement are used for different purposes.

Scale includes understanding that different characteristics, properties, or relationships with a system might change as its dimensions are increased or decreased.

Rate involves comparing one measured quantity with another measured quantity, for example, 60 meters per second.

Robots rely on the use of many innate constants in their basic operation. Ultrasonic sensors, for instance, calculate distance based around an assumed value for the speed of sound.

In calculating the distance a robot travels per spin of its motor, fundamental mathematical relationships govern the elements of change and constancy between the different factors involved. For example, the ratio between the diameter and circumference of the wheel is constant ($C=\pi d$). On the other hand, a robot doesn't always need to use the same wheels – they can change – yet, no matter what the size of the wheel, the distance traveled per turn of the wheel remains proportional.

Measurement is fundamental to all aspects of robotics, from matching dimensions of parts to ensure that they can connect properly, to measuring how far your robot went, to measuring how well a prediction matched a result.

© 2009 Carnegie Mellon Robotics Academy

Science Standards continued

From the National Science Education Standards (NSES)

Evolution and Equilibrium

Evolution is a series of changes, sometimes gradual and sporadic, that accounts for the present form and function of objects, natural systems and designed systems. The general idea of evolution is that the present arises from materials and forms of the past.

Equilibrium is a physical state in which forces and changes occur in opposite and off-setting directions. For example, opposite forces are of the same magnitude, or off-setting changes occur at equal rates.

Every robot design has a story. As they build and modify their robot designs, students can trace the evolution of their creation as they adapt it in different ways that allow it to complete different tasks, building upon lessons learned from their previous designs.

Equilibrium appears in many different forms as a design factor that students will encounter in designing their robots. For example, a robot's top speed is an equilibrium point between the physical force of friction and the force generated by the motor.

Form and Function

Form and function are complementary aspects of objects, organisms, and systems in the natural and designed world.

When designing robots, form always follows function.

Whether the design decision involves using large versus small wheels, making the motor power high versus low, or selecting the sensing device the robot will use, all decisions are based on what the robot is expected to do: its function. All of these decisions will affect the final shape of the robot: its form.

Science as Inquiry – Content Standard "A"

As a result of activities in all grades, all students should develop:

- Abilities necessary to do scientific inquiry
- Understanding about scientific inquiry

Students should be engaged in activities that:

- Begin with a question
- Allow them to perform an investigation
- Gather evidence
- Formulate an answer to the original question
- Communicate the investigative process and results

The guided investigations in Robotics Engineering are targeted at specific relevant questions about robotics technologies and concepts that lead to rich exploratory experiences.

Some investigations focus on specific portions of the inquiry process, such as evidence-gathering or hypothesis evaluation. Others begin with a question and seek an answer using general inquiry processes.

Explanation and evaluation are primary abilities applied in answering questions, not simply calculations or summarization.

© 2009 Carnegie Mellon Robotics Academy

Science Standards continued

From the National Science Education Standards (NSES)

Physical Science – Content Standard "B"

As a result of activities in all grades, all students should develop an understanding of:

- Properties and changes of properties in matter
- Motions and forces
- Transfer of energy

By using simple objects, such as rolling balls and mechanical toys, students can move from qualitative to quantitative descriptions of moving objects and begin to describe the forces acting on the objects.

Understanding of energy will include light, heat, sound, electricity, magnetism, and the motion of objects.

Robotics is able to demonstrate many applied physical concepts. Here are a few examples:

- Mechanical advantage (gears)
- Basic circuitry (sensor operation)
- Digital and analog electronics (sensors)
- Light (lamp, light sensor)
- Sound (ultrasonic, sound sensors)
- Speed (motors)
- Friction (robot movement)

Quantitative measurement is a staple of all investigations.

Science and Technology – Content Standard "E"

As a result of activities in all grades, all students should develop:

- Abilities in technological design
- Understandings about science and technology

Students should begin to differentiate between science and technology.

In the middle school years, scientific investigations can be completed by activities in which the purpose is to meet a human need, solve a problem, or develop a product rather than explore ideas about the natural world.

Robotics is the premier example of the marriage of science and technology, especially as related to the solving of problems or human needs.

Every investigation students conduct with the robot is motivated by the need to advance the performance of the robot in order to meet performance criteria, connecting the "need to know" with the "ability to do".

© 2009 Carnegie Mellon Robotics Academy

Mathematics Standards Addressed

From the National Council of Teachers of Mathematics (NCTM)

Numbers and Operations

- Understand numbers, ways of representing number, relationships among numbers and number systems.
- Understand meaning of operations and how they relate to one another.
- Compute fluently and make reasonable estimates.

Robotics uses numbers and operations in nearly all lessons, for example:

- Calculating distance with rotational sensors (equations, equalities)
- Gears, gear ratios and speed (ratios and proportions)
- Light sensors and threshold (inequalities)
- Wheel circumference, radius and diameter (geometric relationships)

Algebra

- Represent and analyze mathematical situations and structures using algebraic symbols.
- Use mathematical models to represent and understand qualitative relationships.
- Analyze change in various contexts.

Robotics lessons that involve algebra include the following:

- Conditional statements (inequalities)
- Programming sensors and thresholds (inequalities)
- Measuring turns (equalities, solving equations)
- Gears and speed (ratios, direct and indirect proportionality)
- Passing parameters in functions

Geometry

- Precisely describe, classify, and understand relationships among types of two and three-dimensional objects using their defining properties.
- Specify location and describe spatial relation ships using coordinate geometry and other representational systems.

Robotics situations involving geometry include:

- Wheel rotations and circumference (diameter, circumference)
- Identifying locations in order to program a robot to move from point to point (connected path segments)
- Interlocking gears and gear ratios (discrete combinations of radii)

Measurement

- Understand measurable attributes of objects and the units, systems, and processes of measurement.
- Apply appropriate techniques, tools and formulas to determine measurements.

Understanding the significance and meaning of measurements are central to the understanding of robotics:

- Distance the robot travels (linear measurement, meter stick)
- Amount a motor turns (angular measurement)
- Directional change of the robot (angular measurement, protractor)
- Speed of the robot (rate measurement, meter stick, built-in timer)
- Physical quantities measured by sensors (touch, sound, light, distance)
- Detectable region of a sensor (ultrasonic sensor, meter stick, 2D graph paper)

© 2009 Carnegie Mellon Robotics Academy

Mathematics Standards continued

From the National Council of Teachers of Mathematics (NCTM)

Problem Solving

- Build new mathematical knowledge through problem solving.
- Solve problems that arise in mathematics and other contexts.
- Apply and adapt a variety of appropriate strategies to solve problems.
- Monitor and reflect on the process of problem solving.

In the lessons, there are both guided and open-ended design problems that involve designing, building, and programming needed to create autonomous robots.

- How do I get a robot to move a certain distance? (solved through measurement and the verification and use of a proportionality relationship)
- What does the sound sensor measure? (solved by graphing the sensor readings with tones of varying volume and pitch, then seeing which one indicated an orderly relationship)

Reasoning and Proof

- Recognize reasoning and proof as fundamental aspects of mathematics.
- Make and investigate mathematical conjectures.
- Develop and evaluate mathematical arguments and proofs.
- Select and use various types of reasoning and methods of proof.

Reasoning in robotics comes in many different forms, including the following:

- Experimental reasoning, proof using measurements and physical evidence (Wheels and Distance)
- Reasoning using equations, proof by solving (Measured Turns)
- Reasoning about graphs, proof by observing trends (Frequency and Amplitude)

Communications

- Organize and consolidate their mathematical thinking through communications.
- Communicate their mathematical thinking coherently and clearly to peers, teachers, and others.
- Use the language of mathematics to express mathematical ideas precisely.

Activities and Engineering Labs requires documentation that allows student to reflect on what they have accomplished or experienced, and describe it or some aspect of it in their own words to someone else. Emphasis is placed upon explaining reasoning in addition to showing calculations.

The Engineering Design Challenge includes opportunities for students to communicate with their peers and teachers what they have learned and accomplished.

Connections

- Recognize and use connections among mathematical ideas.
- Understand how mathematical ideas interconnect and build on one another to produce a coherent whole.
- Recognize and apply mathematics in contexts outside of mathematics.

One of the strongest features of using robotics to teach math, science, engineering, technology and communications is its ability to make links between multiple disciplines. Students are able to take what they know and connect it to what they are learning, synthesizing new knowledge as they continue.

© 2009 Carnegie Mellon Robotics Academy

Technology Standards Addressed

From the International Technology Education Association (ITEA)

The Nature of Technology

1. Students will develop an understanding of the characteristics and scope of technology. 2. Students will develop an understanding of the core concepts of technology. 3. Students will develop and understanding of the relationships among technologies and the connections between technology and other fields of study.	All robotics activities provide excellent hands-on exposure to technology in use and development. • Robotics activities feature linkages to advanced technologies that allow students to connect their designs to real-world needs and solutions • Successful robot operation revolves around the application of systems concepts to make sensors, actuators, and other components work together • Design processes take into account goals, resources, and trade-off factors to achieve optimal results • Technology exists in proper context alongside applications in science, math, and engineering • Several different technologies (e.g. desktop computer, USB/Bluetooth peripheral interface, mobile robotics controller, electromechanical sensors and actuators) are routinely used together in the operation of the MINDSTORMS robot system, and all are necessary for it to work

Technology and Society

6. Students will develop an understanding of the role of society in the development and use of technology.	Robotics Engineering Design Challenges are linked to real world problems that use similar technologies to accomplish tasks that fulfill a social and/or economic need in the real world. For example: • For instance the robot mining project that is part of this curriculum addresses a real problem that mining industries face daily. Some robot activities focus specifically on Human-Robot Interaction (HRI), an emerging field dealing specifically with psychological and design issues relating to the use of robots in human environments.

Design

8. Students will develop an understanding of the attributes of design 9. Students will develop and understanding of engineering design 10. Students will develop an understanding of the role of troubleshooting, research and development, invention and innovation, and experimentation in problem-solving.	Students gain first-hand experience with developing a functional robotic system in many activities, including: • The Warehouse Programming Challenge • The Robot Mining Challenge • Teacher assigned robot problems

© 2009 Carnegie Mellon Robotics Academy

Technology Standards continued

From the International Technology Education Association (ITEA)

Abilities for a Technological World

11. Students will develop the ability to apply the design process 12. Students will develop the ability to use and maintain technological products and systems	Students will apply design processes continually while working with and developing the robot. Here are some basic examples: • Solving engineering design problems • Robotics Competitions In the course of working with the robot, students will be responsible for the maintenance of their robots: • Mechanical soundness (the robot needs to be kept in good enough condition to perform its tasks daily) • Organizing information (students must keep good records to know how to use systems they initially designed days or weeks earlier) • Troubleshooting (robots have problems–often–and students must be able to identify and solve these issues as they arise) Students will work with many important technologies as part of the operation of the NXT system: • Electronic microcontrollers (NXT) • Desktop/laptop computer and software (NXT Programming Software, word processor for write ups, spreadsheets for data graphs) • Peripheral interfaces (USB or Bluetooth wireless) • Electromechanical systems (touch, light, rotation, sound, ultrasonic sensors) • Electromechanical actuators (Interactive Servo Motors)

The Designed World

16. Students will develop an understanding of and be able to select and use energy and power technologies 17. Students will develop an understanding of and be able to select and use information and communications technologies 18. Students will develop an understanding of and be able to select and use transportation technologies 19. Students will develop an understanding of and be able to select and use manufacturing technologies	The NXT robot is an excellent example and integrator of many different designed technologies working together as a coordinated system. • Power sources (battery technologies – rechargeable Lithium-Ion vs. disposable alkaline) • Vehicle systems (all the robot's systems must work together in order to make it mobile, a viable platform for transportation of goods or as a platform to perform other work) • Manufacturing and prototyping (robot must be built and modified using appropriate materials, plans and tools) • Structural soundness and stability concepts are integral to the design of the robot's physical form. • Communication between system components (desktop to NXT, sensors to NXT, NXT to motors) • Communication technologies (USB vs. Bluetooth)

© 2009 Carnegie Mellon Robotics Academy

Reading, Writing, Listening, and Presenting

Communications skills developed through robotics

Engineering does not exist in a vacuum; it is highly interdisciplinary and highly social. Teamwork is a central foundation of Engineering, and communication is essential to smooth functioning of any engineering team. Students will find that highly developed communication skills are an absolute necessity for success.

Communications skills applied when working with Robots

Situation or Activity	Communications Concepts Applied
Maintain Engineering Design Notebook	Organization of information
Reach consensus on which of several student-proposed designs the team will build	Teamwork and group communication skills • Running and participating in meetings • Building consensus
Compose a compelling proposal to convince a (virtual) sponsor that their robot's development is worth funding	Formal persuasive composition Integrate self-conducted research into a piece that is not purely expository Technical writing • Explaining technical decisions and implementations to an audience that is not necessarily technically inclined
Document the team's progress and accomplishments daily	Documentation and accounting for time, resources, and progress
Undergo review and integrate feedback from experts	Review and feedback processes Learning to accept and respond to criticism
Choose from a variety of representations to best illustrate and communicate a point	Use many different formats of both technical and non-technical information, across different media: • Graphs • Charts • Tables/Matrices • Photographs • Sketches • Timelines • PERT and Gantt Charts • Multimedia presentation • Text
Use Bluetooth to communicate between two NXTs	Electronic communication paradigms (as opposed to human) Networking and connected devices
Establish a standard "language" for communicating between two NXTs	Electronic communication Basic principles of communication • Necessity of shared language • Encoding and interpretation
Programming the robot	Communicate instructions explicitly to a robot using a "foreign" language

© 2009 Carnegie Mellon Robotics Academy

Reading, Writing, Listening, and Presenting continued

Communications skills developed through robotics

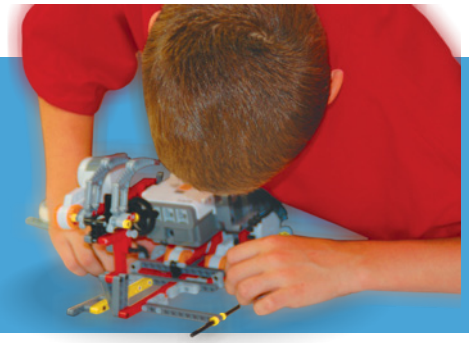
Communications skills applied when working with Robots

Situation or Activity	Communications Concepts Applied
Various interim deliverables intended for either internal or external use	Examples • Descriptive/Explanatory Composition: Describe behaviors, verbalize the functionality of parts of the program • Expository writing: How the machine works • Persuasive/Explanatory Composition: Justify a design choice • Record data in a table, evaluation of methods, predictions, describing robot behavior, describing a proportional relationship • Verbalize troubleshooting processes, analyzing and describing an unexpected situation or observation • Describe a design concept • Compare/contrast design choices, document and record steps, explain why the group took a certain approach • Research, examine and evaluate real-world robot applications • Describe a complex programming concept • Develop a marketing plan for a robot technology

© 2009 Carnegie Mellon Robotics Academy

The Carnegie Mellon Robotics Academy develops...

Classroom-proven tools for teachers that foreground STEM concepts through *hands-on minds-on* robotics activities



Robotics Engineering Vol. I
Introduction to Mobile Robotics CD-Rom

A standards-mapped, STEM-based curriculum with engaging activities, extensive resources, and complete teacher support...

- 18 lessons supported by video tutorials
- Building instructions & programming assistance
- Explanatory animations for each sensor
- Worksheets and data tables for each lesson
- Teacher notes and implementation suggestions
- Handouts and introductory Powerpoint presentations
- Quizzes, answer keys and evaluation rubrics
- Aligns to Math, Science, and Technology standards

Robotics Engineering Vol. II
Guided Research CD-Rom / DVD

A learning continuum (with Vol. I) that features research, teamwork, and 'real world' engineering problem solving...

- Automated mining, patrol robot, and automated tree measurer projects where students learn advanced programming concepts
- 3 guided research and engineering challenges that build on the projects: mine mapping, creating a sentry system, & tree surveying
- Advanced programming videos on loops, switch blocks, data hubs, displaying real time data, storing variables, and Bluetooth
- Teacher notes & implementation suggestions, lesson handouts, evaluation rubrics, quiz answers. Aligns to Math, Science, Technology, and Communications standards

NXT Video Trainer 2.0 DVD

A self-paced guide for students & educators that teaches software programming for NXT-G through basics & beyond...

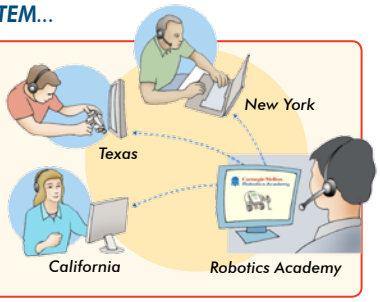
- Focused on introductory programming including motors, sensors, and decision-making
- Self-guided video lesson structure with regular 'check your understanding' questions
- Programming lessons paired with STEM investigations
- Classroom-ready with printable worksheets, teacher guide, and step-by-step video directions

Online Professional Development

A live online course that teaches you how to program NXT-based robots and how to use robotics as an organizer to teach STEM...

Course includes:

- USB headset (yours to keep afterward)
- NXT Video Trainer CD with classroom license ($225 value)
- Online access to supplemental lessons
- Technical support
- Live access to instructors
- 24/7 access to class forums / message boards

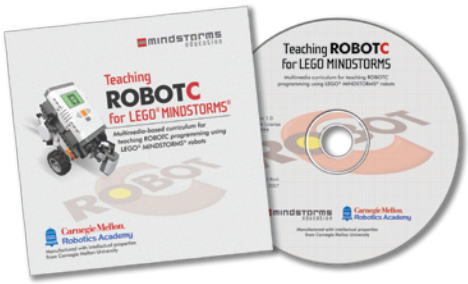


Student learning that contextualizes math, science, technology and engineering

ROBOTC Software

ROBOTC software
Specifically designed to program educational robots. Includes a user-friendly interface with basic and advanced programming options. Based on industry-standard C code and compatible with multiple robot hardware platforms.

ROBOTC Curriculum



Teaching ROBOTC for LEGO MINDSTORMS
Curriculum includes over 45 short videos, over 300 pages of documentation, 20 programming challenges, and quizzes to check student understanding. Leads new programmers step-by-step into the world of industry-standard C-programming.

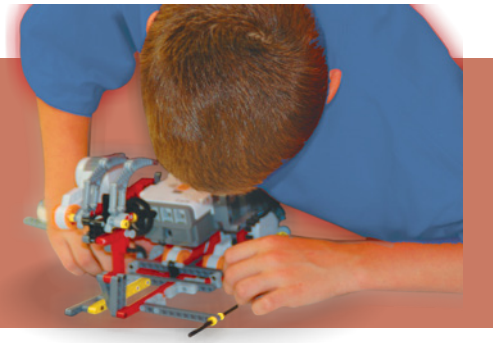
Robotics Curriculum in over 6000 schools...

Carnegie Mellon Robotics Academy

www.education.rec.ri.cmu.edu • 412 681.7160 • Ten 40th St., Pittsburgh, PA 15201

The Carnegie Mellon Robotics Academy develops...

Classroom-proven tools for teachers that foreground STEM concepts through *hands-on minds-on* robotics activities

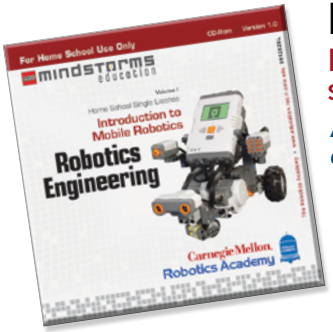


ROBOTC Curriculum

For TETRIX and LEGO MINDSTORMS CD-Rom

A multimedia curriculum that leads new programmers step-by-step into the world of industry-standard 'C' programming...

- Over 50 short videos help new users 'out of the starting gate'.
- Set-up section covers firmware, building your first bot, & more
- Students learn the role of a programmer, and what syntax is
- Lessons on autonomous control of a robot's speed & direction
- Challenges augment lessons with engaging scenarios
- Extensive coverage of sensor hardware and feedback
- Students learn how to use variables and create functions
- Aligns to Math, Science, and Technology standards



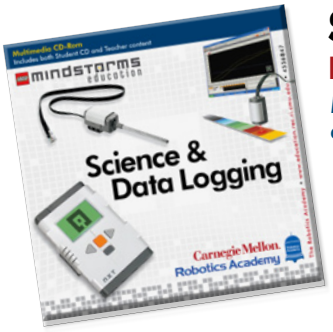
Robotics Engineering Vol. I

Home School CD-Rom

Single License for Home School use only

A standards-mapped, STEM-based curriculum with engaging activities, extensive resources, and complete teacher support...

- 18 lessons supported by video tutorials
- Building instructions & programming assistance
- Explanatory animations for each sensor
- Worksheets and data tables for each lesson
- Teacher notes and implementation suggestions
- Handouts and introductory Powerpoint presentations
- Quizzes, answer keys and evaluation rubrics



Science & Datalogging

Investigations & Experiments CD-Rom

Introduces new programmers & scientists to the data logging capabilities of the NXT, including the temperature sensor...

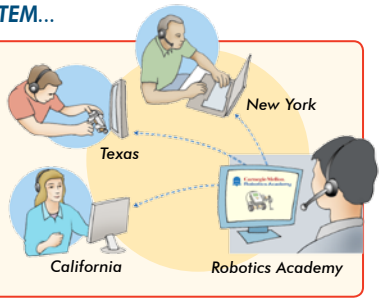
- Six inquiry-focused STEM lessons and projects
- Teacher materials included for each lesson
- Over 40 short videos
- Accompanying worksheets, quizzes, and checks for understanding
- Building instructions for all programs
- Models designed to be built quickly by students in the classroom

Online Professional Development

A live online course that teaches you how to program NXT-based robots and how to use robotics as an organizer to teach STEM...

Course includes:

- USB headset (yours to keep afterward)
- NXT Video Trainer CD with classroom license ($225 value)
- Online access to supplemental lessons
- Technical support
- Live access to instructors
- 24/7 access to class forums / message boards



NXT

Robocamp-on-a-Disk

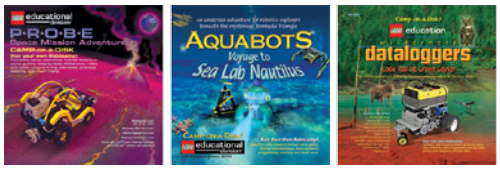
Deep Space Terraformers

Terraformers Camp-on-a-Disk CD takes campers on a mission to make a distant planet habitable for humans. Videos and animations help campers build robots, learn programming, tackle challenges, and hone their robotic and project planning skills.

Terraformers is perfect for a one week robotics camp or a 9-to-12 week after-school program. It provides extensive resources for a Robocamp director, including a step-by-step camp guide, registration forms, awards sheets, gameboard plans, themed props, and more.

RCX

Robocamps-on-a-Disk



Space, Oceans, Forests...

RCX Camp-on-a-Disk products are designed for the legacy RCX brick. They feature extensive resources including a Camp Guide that covers everything from planning to graduation, with recommended day by day activities. Prior experience teaching robotics or programming not required or assumed.

Robotics Curriculum in over 6000 schools...

Carnegie Mellon Robotics Academy

www.education.rec.ri.cmu.edu • 412 681.7160 • Ten 40th St., Pittsburgh, PA 15201