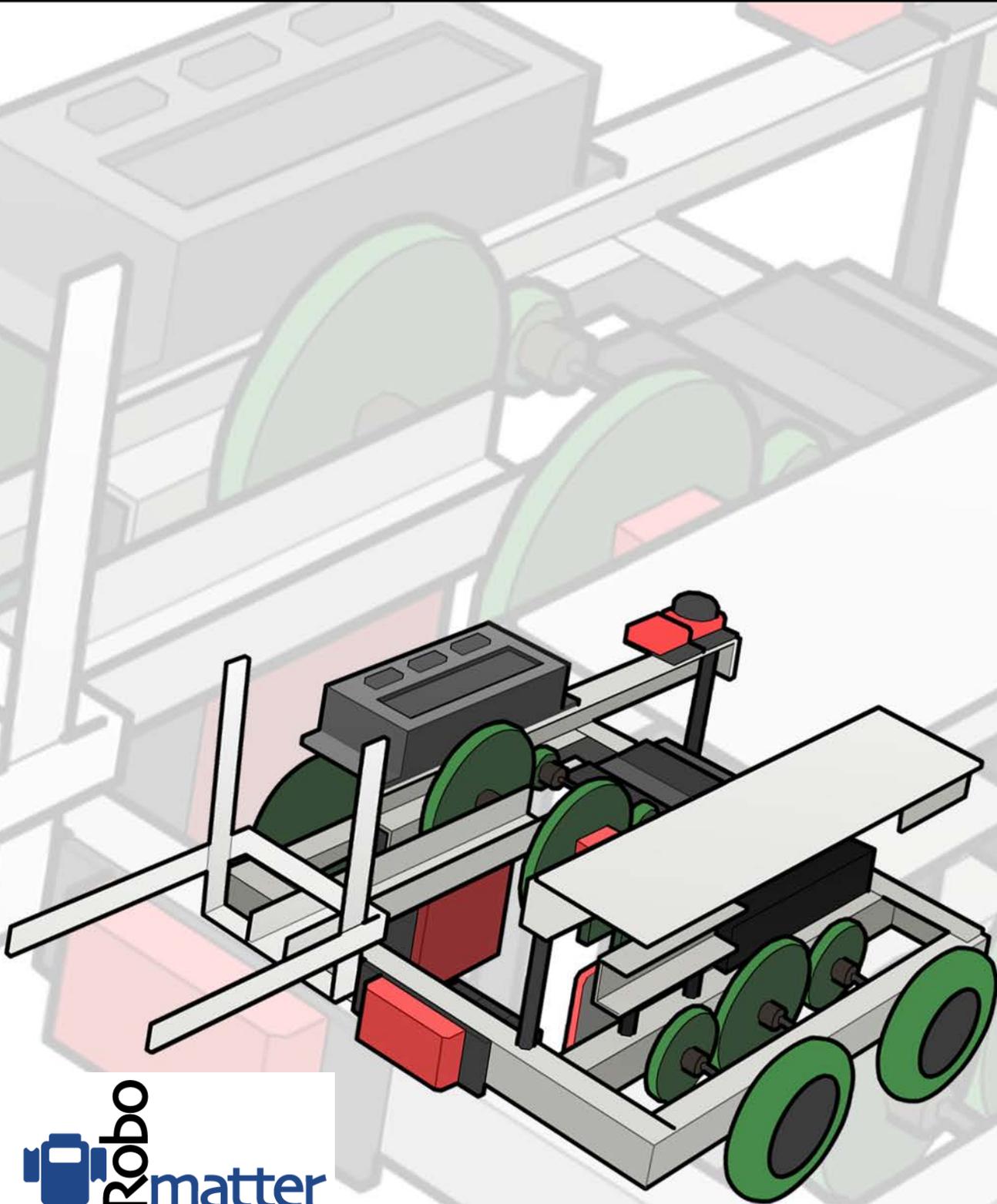
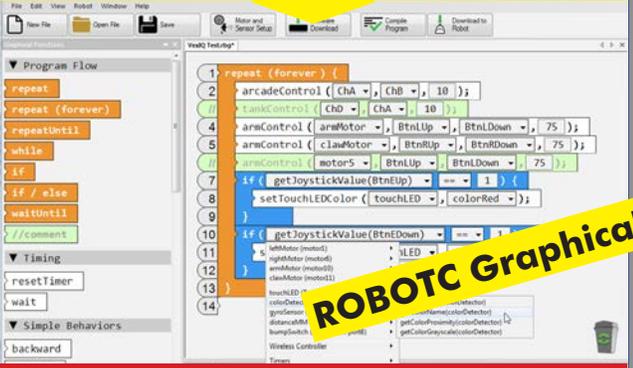


Teacher's Guide



ROBOTC Graphical/ROBOTC Text

Beginners start here...



ROBOTC Graphical

and quickly get here!

```

41
42
43 task main()
44 {
45 //Threshold for line t
46 int threshold = 85;
47
48 //Step 1
49 //Clear the encoder
50 nMotorEncoder
51 nMotorEncoder
52
53 //While the robot
54 //While the robot encoder i
55 //While the robot encoder i
56 //While the robot encoder i
57 motor[rightMotor] = 75;
58 motor[leftMotor] = 75;
59
60
    
```

Standard ROBOTC

ROBOTC
4.0

ROBOTC 4

The Language of Robot Innovation!

ROBOTC 4.0 includes a drag and drop graphical interface that provides a quick way for teachers to introduce students to basic programming logic.

ROBOTC graphical is not intended to replace ROBOTC, but to provide a scaffolded on-ramp to begin to teach programming.

Robot Virtual World Software

Simulation Software that makes a difference!

Robot Virtual Worlds (RVW) has evolved into a game changer in the world of robotics education. RVW enables every student to have their own robot to learn and practice programming. Students build their program in simulation and then test it on a physical robot. Homework packs are priced at less than \$5 per student.

Test your code with a virtual robot then use the same code on your physical robot!



Work on code for a VEX competition, new competitions are modeled each year

www.robotvirtualworlds.com



or have kids learn to write code using these fantasy challenges.

Teaching Robotics! What are you going to teach?

If you are new to robotics, or are designing your robotics course it is important to think about what you want to teach. Yes, you are going to teach robotics, but what is it that your course will emphasize? Robotics Academy research shows that any one robotics lesson can cover five or more different concepts, *and* unless you foreground what you want students to learn, that little or no “new measurable learning” will take place. Think about the following:

- Robotics is an eclectic mix of applied science, mathematics, computer science, mechanics, physics, engineering design, sensors, problem solving... what will your course emphasize?
- When kids study robotics they can learn to develop 21st century skills like: cooperation and collaboration, teamwork and problem solving, critical thinking and creativity... Are these skills important to you?
- When students engage in robotics engineering activities they can learn important lessons like: the first step to solving a problem is to conduct research, how to allocate and manage time, the need to develop a set of plans before you start building, the iterative nature of design, how to work with and manage people... What activities will use in your course to teach these lessons?
- Each student that comes into your classroom has a different world experience that they draw from. It is important to scaffold instruction but you don't want to limit creativity. How can your course support students with varying technical backgrounds?

General topics that robotics can teach

Robotics is a relatively new to education, and teachers and administrators have varying reasons why they want to implement a robotics program; we've learned that there is no “one size fits all”. As you are building your robotics program, you will want to carefully consider what it is that you want to teach via robotics. Here are a couple of ideas:

- Grade level mathematics - there are many ways to mathematize robots via programming; computing distances, angle, and unit rates, passing parameters in the form of formulas, and applying geometry trigonometry, and physics to a student robot design.
- Reading and writing with understanding - There are many ways to incorporate research, writing, and presentations into robotics courses. When students engage in an engineering design problem they should begin by researching how others have solved the problem in the past. From there they write up what they've learned and if time allows make a presentation. All of these skills tie into the English and Language Arts College and Career Readiness Standards.
- Introductory through advanced programming concepts - Computational thinking and programming are new basics that students must understand. See the K-12 Computer Science Teacher Association Standards and you will find that robotics addresses many of those standards.
- Introduction to engineering - Robotics provides many opportunities for students to apply engineering design principles to solve complex problems that require them to consider trade-offs such as costs, safety, reliability, and aesthetics; they are learning engineering. These skills can be taught at the elementary through college level and aligns with the new Science and Technology standards.
- Teaching technological literacy - Robotic systems require student to engage with computers, sensors, the Internet, electrical mechanical actuators, and a plethora of evolving technologies.
- The development of 21st Century Skills - There are over 35,000 US school based robotics teams. Robotics competitions are purposefully designed to place students in situations where they have to work in teams and problem solve. Properly designed activities develop 21st century skills in students.

Table of Contents

Table of Contents

- 3 Things to Consider When Teaching Robotics
- 5 FAQ
- 6 Checklist/How do I use this teacher's guide?

7 Introduction

- 7 VEX Video Trainer Lesson Structure
- 8 Classroom Setup & System Requirements
- 9 What does the VEX Cortex Video Trainer Teach/Differentiated Instruction
- 10 How do I use the Curriculum in my classroom?
- 11 What topics are covered in each Unit?
- 12-13 General Layout of the Curriculum

14 Scope and Sequence

- 14 Class Rules and Organization
First Assignment
Safety
The Rube Goldberg Machine™
- 15 Your First Robot
Teaching Robot Math
Teaching Programming Using Simulation Software
The Movement Chapter
- 16 The Remote Control Chapter
The Sensing Chapter
The Engineering Chapter

17 Chapter Overview

- 17 The Fundamentals Chapter
- 18 The Setup Chapter
- 19 The Movement Chapter
- 25 The Remote Control Chapter
- 27 The Sensing Chapter
- 32 The Engineering Chapter

35 Navigating the Curriculum

- 35 The Fundamentals Chapter
- 44 The Setup Chapter
- 56 The Movement Chapter
- 68 The Remote Control Chapter
- 74 The Sensing Chapter
- 93 The Engineering Chapter

Important Resources!

- 94 Safety
- 112 Breaking Programs into Behaviors
- 113 Sense Plan Act
- 114 Teaching Pseudocode & Flowcharts
- 115-119 Introduction to Pseudocode
- 120-122 Introduction to Flowcharts
- 124 Introduction to Robot Programming
- 125 Teaching How to Troubleshoot Programs
- 45-46 Building Your Robot
- 33 & 98 Engineering Process
- 100 Project Planning
- 102 Assessment Rubrics

Online Resources

- 104 www.vexteacher.com
- 105 Teaching Engineering Design Process
- 107 Planning Your Project: Technical Sketching
- 110 VEX Robotics Competitions
- 110-111 VEX In-School Design Problems

Frequently Asked Questions

Before starting

- ▶ **Will *the VEX Cortex Video Trainer* help me teach to Standards?**
Yes! See Appendix
- ▶ **What do I need to prepare for class?**
See Checklist, page 6; Lesson Structure, page 7; and Classroom Setup page 8.
- ▶ **What general topics are covered with the curriculum?**
See Topics Covered, page 9 and 11 and the Chapter Overview on pages 17-34.
- ▶ **How do I use the VEX Cortex Video Trainer in the classroom?**
See Using the VEX Cortex Video Trainer in Class, page 10.
- ▶ **I want to know what's in each Chapter and Unit, where to I go?**
See Chapter Overview Fundamentals page 17; Setup pages 18-19; movement, pages 19 - 24; Remote Control, pages 25 - 26; Sensing pages 27 - 31; and Engineering pages 32 - 34.

During class

- ▶ **How do I teach kids to think about programming?**
See "Breaking Programs into Behaviors" and "Sense Plan Act" pages 112 - 113.
- ▶ **What should I teach and when should I teach it?**
See the Scope and Sequence section, pages 17 - 34.
- ▶ **What do I do about students who go faster/slower than the others?**
See Differentiated Instruction, page 9.

After class

- ▶ **How do I prepare my kids for competitions?**
See Robotics Competitions: see the Engineering Section pages 32 - 34 and pages 98 - 101.
- ▶ **Are there quizzes or homework?**
Each sub unit page includes a series of "Check your understanding" questions that can be used to develop a unit quiz. See Rubrics, pages 102 - 103. Robots are hard to take home, but there are many ways to incorporate research assignments or Robot Virtual World assignments into homework assignments.

Checklist/How do I use this teacher's guide?

The curriculum guide is organized in four sections:

Introduction - pages 7-13 - background information to get started

Scope and Sequence - pages 14-16 - a short description of the curriculum

Chapter Overview - pages 17-34 - a detailed description of the curriculum

Navigating the Curriculum - pages 35-102 - a pictorial description of the curriculum

- Identify the Goals of your Robotics Course**
Robotics can be used to teach to lots of standards. This curriculum is designed to introduce students to how to program, an important part of robotics, but not the only thing that you can teach through robotics. Determine what you want students to learn in your class. See page 3.
- Set up the student workstations**
See page 8, Workstation Setup.
- (Recommended) Build the Clawbot with Sensors**
VEX kit configuration has changed multiple times over the years. There are multiple robot build plans in the Setup section of the curriculum, we recommend that your standard build is Clawbot. We also recommend that if your budget allows it, that you have one set of robots to teach programming and another set of robot parts to teach engineering and for competitions.
- Become familiar with the lessons**
See page 7 to become familiar with the lesson flow. Read the Scope and Sequence, pages 14-16; the Chapter Overview, pages 17 - 34, where each chapter is described in a shortened format, and Navigating the Curriculum, pages 35 - 103, where you will find a pictorial view of what you will find on each page of the VEX Cortex Video Trainer.
- Determine overall pacing for the module**
Identify key dates that you would like to have each project due by; make these clear to students in your syllabus or assignment sheets.
- (Highly Recommended) Read the following lessons on how to teach students about how to think about writing code for robots.**
See Breaking Programs into Behaviors, page 112; Sense Plan Act, page 113; Introduction to Pseudocode, page 114; and Introduction to Flowcharts, page 120.
- (Highly Recommended)** Get involved with the RECF Foundation's annual VEX competition challenge. The challenge changes every year and provides teachers with a brilliant engineering problem for their class to solve each year.

VEX Video Trainer Lesson Structure

Guided Robot Programming Activities, Extension Activities for Advanced Students, and Engineering Investigations

The *VEX Cortex Video Trainer Curriculum* is designed to teach introductory level robot programming, logic and reasoning skills, and engineering process using robotics at the context. The core curriculum consists of four chapters (Movement, Remote Control, Sensing, and Engineering) and each chapter is broken into units that teach key robotics programming and engineering concepts. Additionally, there is a huge amount of support for teachers coaching teams in Robotics Competitions for the first time; see the engineering section of the curriculum.

Each unit comprises a unit level programming challenge that students will to solve by the end of the unit.

- ▶ **ROBOTC RBC Files** RBC files are starter programs that will automatically open once your software is installed and configured, click the file and the starter program will open. Some browsers will require uses to save and then open the file using ROBOTC software.
- ▶ **Additional Unit Level Robotic Programming Challenges** and setup guides
- ▶ **Step-by-step guided video instruction** that introduces key lesson concepts (e.g. Loops). Students are taught programming using a step-by-step process; foundational programming concepts are integrated into each unit and repeated in subsequent units.
- ▶ **Built-in “Check your understanding” questions** designed to provide students with instant feedback on whether they understood the big ideas in each lesson.
- ▶ **Reference Guides** that are designed to support the lesson (e.g. white space, comments, loops, conditional statements, Boolean logic, etc.)
- ▶ **Engineering Investigations** that guide students through an engineering experiment, the activities are designed to deepen students’ understanding of that concept.
- ▶ **Robot Virtual World** extension activities. The RVW activities are designed to significantly enhance student’s programming opportunities allowing them to program robots underwater, on an island, in outer space, and via a VEX Cortex international competition. Students are able to use the same programming commands on their virtual solution as they do on their physical robot.

Classroom Setup

What is the best setup for student workstations?

Ideally, pairs of students will work together at one computer, with one VEX robot.

Set up each workstation with:

- a. **ROBOTC 4 for VEX Software** installed on each computer.
 - Check each computer to see that the software works
 - Check each computer to see that Robot Virtual World software works
- b. Access to the **The VEX Cortex Video Trainer Curriculum** software
 - This can be installed locally or on a local network server with proper licensing
 - This may also be accessed remotely via Internet, if your school's network infrastructure and policies allow
- c. Two pairs of **headphones** with **headphone splitters**
 - One pair for each student
 - Avoid using speakers, as multiple workstations in the same classroom will generate too much overlapping noise
- d. One **VEX Cortex robot kit per work station**
- e. **ROBOT Virtual World Software** This software is not required to use the curriculum and complete the lessons, but research shows that it is a very effective tool to teach programming.

What are the System Requirements for the VEX Cortex Video Trainer Curriculum?

Introduction to Programming VEX Curriculum

- HTML5-compatible browser (Firefox, Chrome, Internet Explorer 10+)
- Tablets (iPad, Android, Windows) with HTML5 browsers should work as well when accessing the curriculum from the Internet

Robot Virtual World Software

- PC Compatible OS: with an Intel Core 2 processor family or better
- Memory: 2 GB RAM
- Graphics: NVIDIA® 8800GTS or better, ATI Radeon™ HD 3850 or better
- Hard Drive: 1.5 GB free hard drive space to install all virtual worlds

What does the VEX Cortex Video Trainer Teach?

- ▶ How to control basic robot movements
 - a. Robot math
 - b. Sequences of commands, structures, computational thinking
- ▶ Sensors and how they work
- ▶ Intermediate concepts of programming
 - a. Program Flow Model
 - b. Programming Remote Controls
 - c. Decision-Making Structures
 - Loops
 - Conditionals
 - Functions
- ▶ Teach troubleshooting strategies and engineering practices
 - a. Problem-solving strategies
 - b. Teamwork
- ▶ The iterative nature of engineering design and process

Differentiated Instruction

One of the biggest challenges facing teachers today is meeting the needs of each individual student in their classroom; that is the core of differentiated instruction. Differentiated instruction asks teachers to approach students at their instructional level, and requires students to show evidence of growth from their instructional level. Differentiated instruction encompasses more than just assessment. It involves all aspects of instruction: classroom delivery, overall learning environment, learning content, and assessment. The VEX Cortex Video Trainer provides many opportunities for students of all abilities:

- ▶ Programming - the unit challenges are supported by step-by-step instructional videos that students can work through at their own pace.
- ▶ Solving the open-ended programming challenges embedded into the units that make up the Movement, Remote Control, Sensing, and Engineering Units.
- ▶ Completing the virtual programming challenges found in the Robot Virtual World games (Ruins of Atlantis, Palm Island, Operation Reset, Highrise): attempt to complete the entire world, or choose to program different robots within a Virtual World.
- ▶ Challenging gifted students to iteratively improve their engineering and programming solutions using ROBOTC.
- ▶ Working cooperatively with students having difficulty grasping some concepts.
- ▶ Engaging in engineering challenges that are found in robotics competitions.

Using the VEX Cortex Video Trainer in Class

The VEX Cortex Video Trainer Curriculum is designed for student self-pacing in small groups, preferably pairs that are working together at one computer, with one VEX Cortex robot. It can also be used in “virtual mode” where students are learning programming using a virtual robot that is programmed using the exact same commands that they will use on their actual VEX Cortex robot.

Programming tasks are designed to involve some – but not extensive – mechanical consideration, so that hands-on design tasks may remain authentic without becoming logistically difficult. The Engineering section enables larger teams and requires more building.

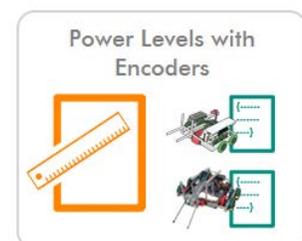
Solutions will not need parts in excess of those included in the VEX Cortex core set, so it is sufficient to leave each team with one kit (although access to additional parts may allow students to construct more creative solutions to problems).

A typical plan for a programming unit (movement, remote control, and sensing) is:

1. Review the unit challenge which is always the first lesson on the page.
2. Groups proceed through the video trainer materials at their own pace, watching the video and then answering the Check Your Understanding questions.
3. At the bottom of many of the pages students will find Engineering Investigations and/or Programming Challenges (see pictures below). They should complete the challenges in the order that they are presented.
4. Each group constructs its own solution to the Unit Challenge
 - Groups may be asked to document their solutions in journals or logs, and especially to explain how they overcame the key problems identified at the start of the unit
5. Assign the Additional assignments based on the focus of the class
 - There are many handouts and challenges that allow teachers to extend the use of the curriculum to teach engineering practices, prepare for robotic competitions, or teach computational thinking generally
 - Complete teacher assigned work.

Engineering Investigations and Programming Challenges

At the bottom of many of the lessons students will find programming challenges and engineering investigations. The Basketball Drills picture shows the icon that indicates a programming challenge, the Power Levels with Encoders icon represents and Engineering Investigation



Topics are Covered in Each Unit

Unit Name	Main Topics
Fundamentals - 10 days	
Introduction to Programming	Basic rules on how to think about programming and syntax
Natural Language Programming	A set of reference documents on Natural Language programming
Setup - 10 days	
Build	Robot Building Instructions
Wireless System Configuration	How to setup and configure VEX Cortex wireless communications
Wired System Configuration	How to configure a wired system
Downloading Sample Programs	How to download a sample program, there are nearly 200 sample programs!
Virtual Robot Configuration	How to use Robot Virtual Worlds
Movement Unit - 20 days	
Moving Forward	Level one programming
Speed and Direction	How to change speed and direction
Shaft Encoders	Loops, configure and use shaft encoders
Automated Straightening	If/Else, variables, and values
Integrated Encoders	How to use IMEs, PID, and precise movement
Remote Control - 15 days	
Joystick Mapping	How to program the VEXnet Joystick
Timers	How to program and use Timers
Buttons	How to program buttons on the VEX Joystick
Sensing - 25 days	
Limit the Arm	The Touch Sensor
Behaviors and Function	Functions and Parameters
Forward Until Near	The Ultrasonic Sensor
Line Tracking	Line Tracking Sensors
Turn for Angle	The Gyro Sensor
Using the LCD	Displaying Messages on the LCD
Engineering - semester	
Safety	Handouts and quizzes
VEX Hardware	Handouts and guides
Engineering Process	Instructional videos and handouts
Competition Programming	Instructional videos and handouts
Rubrics	Assessment rubrics

General Layout of the VEX Cortex Video Trainer Using ROBOTC

The VEX Cortex Video Trainer uses the buttons at the top of the page to navigate the Fundamentals, Setup, Movement, Remote Control, Sensing, and Engineering sections of the curriculum. The last button, Reference, is designed to allow the user to navigate the whole curriculum to find a handout or video lesson.

The image displays six screenshots of the VEX Cortex Video Trainer interface, arranged in a collage. Each screenshot shows a different section of the curriculum, with red lines indicating the navigation flow from the top menu to the content area.

- FUNDAMENTALS:** The top menu is highlighted. The main content area includes an introduction to programming and a list of topics: 1. Introduction to Programming, 2. Natural Language Programming.
- SETUP:** The top menu is highlighted. The main content area includes instructions on installing ROBOTC and a list of steps: 1. Build, 2. Wireless System Configuration, 3. Wired System Configuration, 4. Download Sample, 5. Virtual Robot Configuration.
- MOVEMENT:** The top menu is highlighted. The main content area includes a challenge description and a list of topics: 1. Challenge, 2. Moving Forward, 3. Speed and Direction, 4. Shaft Encoder, 5. Automated Straight, 6. Integrated Encoders.
- REMOTE CONTROL:** The top menu is highlighted. The main content area includes a description of VEXnet Remote Control and a list of topics: 1. Challenge, 2. Joystick Mapping, 3. Timers, 4. Buttons.
- SENSING:** The top menu is highlighted. The main content area includes a description of sensors and a list of topics: 1. Challenge, 2. Limiting the Arm, 3. Behaviors and Functions, 4. Forward until Near, 5. Line Tracking, 6. Turn for Angle, 7. Using the LCD.
- ENGINEERING:** The top menu is highlighted. The main content area includes a description of safety and engineering processes and a list of topics: 1. Safety, 2. VEX Hardware, 3. Engineering Process, 4. Competition Programming, 5. Rubrics.

General layout continued next page.

General Layout continued

The screenshot shows the ROBOTC website interface. At the top is a navigation bar with icons for HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. Below this is a large blue banner for 'FUNDAMENTALS'. The main content area features a table of contents with two main sections: '1. Introduction to Programming' and '2. Natural Language Programming'. A red arrow points from the '1. Introduction to Programming' section to a sub-page example on the right. The sub-page example is titled 'ROBOTC Rules Part 1' and includes a video player, a 'Check Your Understanding' section with five multiple-choice questions, and two reference material icons: 'Whitespaces' and 'Reserved Words'. At the bottom of the sub-page, there are navigation buttons for 'Prev Lesson: Planning and Behaviors' and 'Next Lesson: ROBOTC Rules Part 2'.

ROBOTC Fundamentals Video
Each lesson has an individual video to support students

Check your Understandings
Questions designed to provide students time to reflect on what they just learned

Reference Materials
to support the lesson

Sub-Pages

Each sub page within the main navigation opens to a sub-page. Pictured at the right is the ROBOTC Rules Part 1 sub-page. The majority of the pages contain one short video. As set of “check your understandings” questions that asks students questions about the video that they just watched, and additional reference materials if they are important to the lesson.

Scope and Sequence

This scope and sequence is a short three page version of what a year long class might look like. For a more detailed description of what is in each chapter go to the Chapter Overview section starting on page 17. The outline below uses the following format:

Approximate Class Days	A brief description of the lesson and where to find it in the curriculum.
------------------------	---

Introduction to Robotics Course Outline

Note: there are many ways to teach this course. This set of general guidelines assumes that you want to teach a robotics engineering course and want to begin by teaching programming.

3-5 days

Class Rules and Organization/Assessment/Engineering Journal

The first week of class covers classroom organization and an overview of what students will learn in your class. A key concept that students should have when they leave this course is: "What a robot?". In today's world, robots are everywhere, we just don't call them robots. Begin with the handout on page 15, Sense Plan Act. Have the students identify as many technologies that fit that description as possible and sort them into industry sectors (i.e. banking, manufacturing, entertainment, healthcare...). Have students select an industry sector and answer the following question: How has robotics affected that industry sector? Or assign them the general topic: What is a robot?

First Assignment - "What is a robot?"

3-5 days

Safety Lesson - page 32 and pages 94-95

Safety is an integral part of an robotics course and should be covered at the beginning of the class and then strictly enforced throughout the year. You will find many handouts, worksheets, and quizzes in the Safety Unit found in the Engineering Chapter. Every class setting is different, it is the responsibility of the teacher to select the appropriate handouts and tools to teach safety in their class. It is paramount that students understand that a clean lab is a safe lab and that they are responsible to cleanup at the end of each day.

10 days

Introduction to Engineering (Rube Goldberg Machine™) - Internet

Students will be working on engineering design problems throughout their robotics class. A good first assignment is to randomly break students into teams and have them complete a "Rube Goldberg Machine" (RGM). RBM contests are designed to encourage teamwork and out-of-the-box problem solving, it is also a fun activity that allows you to introduce the engineering process activities that are presented in the Engineering Chapter: Engineering Process, Engineering Design Journals, Teamwork, Project Planning, PERT or Gantt Charts, etc. It will be the up to the teacher to determine what parts students will use for their RGMs (e.g. VEX parts, random parts from home, or a combination of parts) Be sure to set a limit on the number of days that you give students to complete their project. For RGM rules go to www.rubegoldberg.com

The RGM project also provides the teacher with the opportunity to introduce students to the assessment tools that they will use throughout the course.

Rubrics - Introduce students to your evaluation tools

- Engineering Journal Rubric - pages 98-101
- Work Habits Rubric - page 102
- Writing and Proposal Rubrics - page 102

Scope and Sequence Continued

5-10 days

Setup Unit

- **Build Robot - pages 45-46**

If you do not have robots built, then now is the time to build them. There are an unlimited number of robot builds that you can design using VEX kits, we suggest that you use the “Clawbot with Sensors” build. It is important that whatever *teaching robot* that you choose for your class to build that all of the robots have the same configuration (motors are in the same ports, sensors are in the same ports, all robots have the same gripper). If every student’s robot is different, uses different mechanics, has sensors plugged into different ports, and has different types of grippers, then it will take you much longer to help them to troubleshoot their robot’s problems.

Once the Robots are Built Test Them

- Robot Configuration - see pages 47-48
- Download a sample program - see page 49

5 days



Teaching Robot Math - Expedition Atlantis Robot Math Game

Expedition Atlantis is a free math programming game that has proven to teach robot math to students. It is easier to teach the math without having students worry about the programming. You can download the software from the Robot Virtual World website. The software includes an easy to follow teacher guide.

Teaching Programming Using Simulation Software - pages 53-54

Many teachers are turning to Robot Virtual World software to teach the programming portion of the curriculum. Research shows that when students are trying to learn multiple concepts at the same time (programming, mechanics, sensors, and math) that they become confused. It is much easier to foreground and teach one concept at a time rather than try to teach multiple concepts at a time. The Robot Virtual World software eliminates students having to learn the mechanics of the system (How do I configure my robot to talk with VEXnet? What port am I plugged into? Is my robot battery charged? Did someone in another class change things on my robot?). The curriculum can be taught without RVWs, but RVWs is a more efficient way to teach robot programming.

20 days

The Movement Chapter - pages 19-24 and pages 56-67

The Movement Chapter provides a scaffolded way to introduce students to basic programming using ROBOTC and VEX robots. Each Chapter in the curriculum includes a Chapter Challenge; in the Movement Chapter the challenge is the Labyrinth Challenge.

IMPORTANT

Begin the chapter by introducing the challenge and explain to students that they will learn to program the Labyrinth Challenge multiple ways, from using very simple timing to using feedback from encoders, to developing their own automated straightening algorithm, to using PID. The curriculum uses this “simple to efficient” way to introduce programming concepts to students in all of the chapters. For students to truly learn programming, they should complete all methods of solving the challenge.

FUNDAMENTALS CHAPTER

Begin with the “Moving Forward” unit in the Movement Chapter before you spend time in the Fundamentals Chapter. Once students complete the Moving Forward unit they will have context that they can relate to when they learn what is taught in the Fundamentals Chapter. The Fundamental Chapter teaches students: how machines think, what behaviors are, and what comments, whitespace, reserved words, compiler errors, and general rules around syntax are. Students can begin by using pseudocode in the Movement Chapter and flowcharts in the Remote Control and Sensor’s Chapter when the programming logic becomes more complex.

Scope and Sequence Continued

15 days **The Remote Control Chapter - pages 25-26 and pages 68-73**
 The Remote control unit begins with the Minefield Challenge. Once again, the chapter is structured showing simple methods and then advanced methods of programming. This chapter also introduces students to Loops, Boolean Logic, Timers, and programming the VEX remote control buttons to automatically elicit specific behaviors (i.e. press this button and do a 90 degree turn).

Pseudocode and Flowcharts
 It is common for beginning programmers to write programs that compile and that they believe that the logic is correct, but then the robot doesn't do what they think that it should do. Starting with the Remote Control chapter, require students write their program using pseudocode and then develop a flowchart that illustrates the robot's decision making. You will find classroom resources to teach these important processes on pages 17 - 25 of this teacher's guide.

Minefield Competition
 The Minefield Competition is an end of chapter activity that provides a game like environment that is motivating to many students. Feel free to modify the game. Provide opportunities for student to write the rules for the game.

25 days **The Sensing Chapter - pages 27-31 and pages 74-92**

The Grand Challenge!
 The Sensing Chapter begins with the Grand Challenge. The Grand Challenge provides an opportunity for students to demonstrate the ability to use all of the sensors. What is shown on the Grand Challenge video is an example of what the challenge could look like. We recommend that you provide students with a set of behaviors (not the actual code) that their robot will need to complete and that you don't actually share the challenge with them until two days before the actual challenge. For example, your robot will need to be able to complete the following behaviors: Stop at a black line, turn accurately using feedback from a gyro sensor, identify the distance from an object using feedback from an ultrasonic sensor, and identify the location of an object and pick it up and come home.

The Sensing Chapter continues to build on a student's basic understanding of programming and logic. There are many ways to program a robot to complete a task. It is important for the teacher to not only watch the robot perform the task, but also to look at the student's code to make sure that they are applying all of the programming concepts that each individual unit attempts to teach.

The programming unit teaches students how to write functions and pass parameters, about variable types, if/else statements, switch cases, and all of the VEX Cortex sensors. The curriculum continues to use many challenges that the students have seen before, but requires them to solve the challenge using a different, more advanced programming strategy. Feel free to allow the students to modify the programming challenges as long as they teach the same foundational concepts.

Second Semester 90 Days
The Engineering Chapter - pages 32-34 and pages 93-103
 The VEX Cortex Video Trainer is intended to be used to teach students introductory programming. To truly learn engineering, students must be engaged in multiple engineering problems. We strongly suggest that you enroll your students in a Robotics Education and Competition Foundation (RECF) competition. The RECF sponsors a new competition each year and provides teachers with a great tool to teach engineering with. When students are given a problem have them begin by conducting research to see how others solved the problem. Require them to use all of the engineering tools found in the Engineering Chapter: the Engineering Journal, time management tools like PERT and Gantt Charts, and Design Reviews.

Chapter Overview

What topics are covered in each Unit?

Chapter: Fundamentals - 10 days

1. Introduction to Programming - 5 days, then ongoing

- a. Programmer and Machine - an instructional video that introduces the new programmer to how they need to think to translate their ideas into a language that a machine can understand.
- b. Planning and Behaviors - an instructional video that introduces the new programmer to the idea of robot behaviors. This sub-unit also includes three reference PDFs: Behaviors, Pseudocode and Flowcharts, and Thinking about Programming.
- c. ROBOTC Rules Part 1 - an instructional video that introduces students to C programming syntax and how ROBOTC uses color to indicate reserved words. The sub-unit also includes two reference PDFs: Whitespace and Reserved Words.
- d. ROBOTC Rules Part 2 - a followup instructional video that continues to teach students about syntax, comments, and error messages. This sub-unit also includes three reference PDFs: Comments, ROBOTC Error Messages, and ROBOTC Rules.
- e. In this teacher guide you will find several introduction to programming guides:
 - Breaking Programs into behaviors, page 14
 - Sense Plan Act, page 15
 - Introduction to pseudocode, pages 16 - 21
 - Introduction to flowcharts, pages 22 - 25

2. Natural Language Programming

- a. Natural Language programming places code segments into functions and is intended to make it easier to enable new programmers to program simple code. Natural Language provides a bridge between ROBOTC Graphical and full ROBOTC.

It is our recommendation that teachers and students learn to program using full ROBOTC and use the Movement, Remote Control, and Sensing units as they are currently written and supported to teach and learn VEX robot programming.

The Natural Language programming unit contains a large number of reference guides that have been added as a resource for teachers.

Chapter Overview Continued

Chapter: Setup

The Setup Unit contains resources that you and your students can use the first time that they are starting with their robots.

1. Lesson: Build

The build section includes building instructions for four robot types.

- a. Recbot Building Instructions
- b. Squarebot 4 Building Instructions for use with ProtoBot kits
- c. Clawbot with Sensors Building Instructions (***recommended build for this curriculum***)
- d. Swervebot Building Instructions

2. Wireless System Configuration

- a. Update Cortex Firmware (Wireless) - an instructional video that guides the new user how to update the Cortex firmware with the Master CPU Firmware and the ROBOTC Firmware. This unit also includes Check your Understanding questions and a VEX Cortex Driver Installation guide produced by VEX robotics.
- b. Updating VEXnet Joystick Firmware - an instructional video that shows the new user how to update the VEXnet Joystick with the latest firmware using ROBOTC. The lesson also includes Check Your Understanding questions and two PDFs: Establishing a VEXnet Link and VEXnet Joystick Calibration

3. Wired System Configuration

- a. Update Cortex Firmware (Wired) - an instructional video that guides the new user how to update the Cortex firmware with the Master CPU Firmware and the ROBOTC Firmware. This unit also includes Check Your Understanding questions and a VEX Cortex Driver Installation guide produced by VEX robotics.

4. Download Sample Programs

- a. Download a Sample Program Part 1 - An instructional video that shows how to download a sample program over VEXnet. The lesson also includes a USB-to-Serial Cable Driver Installation instructional PDF.
- b. Download a Sample Program Part 2 - The second part of the Download Sample Program video. This lesson also includes Check Your Understanding questions.
- c. Download a Sample Program over USB - and instructional PDF that takes the new programmer step-by-ste through the download sample program over USB procedure.

Continued next page

Chapter Overview Continued

Chapter: Setup Continued

5. Virtual Robot Configuration

- a. Download your First Program (Virtual) - Robot Virtual Worlds (RVW) enable all kids with computers to have access to a robot. This lesson includes an instructional video and Check Your Understanding questions. You can find out more about RVWs at: www.robotvirtualworlds.com
- b. Camera Operation in RVWs - This lesson shows how to use some of the additional controls in the RVW environment. This lesson includes an instructional video and Check Your Understanding questions.
- c. The Measurement Toolkit - This lesson shows a new RVW user how to use the tools found in the measurement toolkit which are embedded into all RVWs. This lesson includes an instructional video and Check Your Understanding questions.

Chapter: Movement Chapter - 20 days

1. Labyrinth Challenge Unit/Movement Chapter

- a. The Labyrinth Challenge - This lesson set is designed to show the challenge that students will work on for their first problem. The Labyrinth Challenge requires students to program their robot to travel a specific distance, turn accurately, and then repeat these behaviors multiple times.

Note: RBC files are used throughout the curriculum. They are designed to automatically start a program when you are ready to begin programming. At the left is the virtual robot icon and you will need to have a working copy of RVWs in order to open that file type. At the right (below) is a physical robot icon. When you select those files and click them ROBOTC will automatically open.



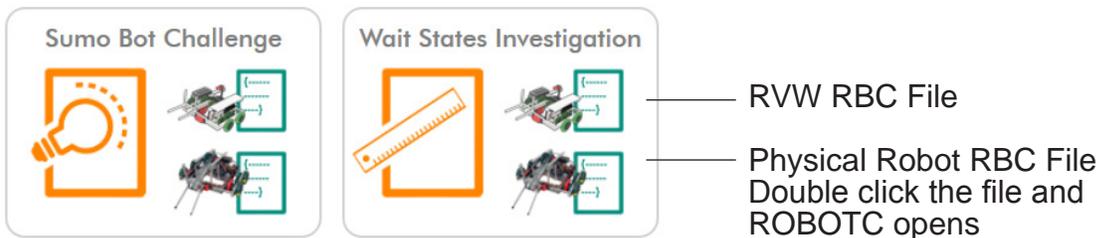
2. Moving Forward Unit/Movement Chapter - 5 days

- a. Program Dissection - This lesson explains what the introductory lines of code do. It also begins to teach some of the syntax related to ROBOTC. This lesson set includes: an instructional video, RBC files, Check Your Understanding questions, and an instructional PDF that walks students through the steps to run a program.

Continued next page

Chapter Overview Continued

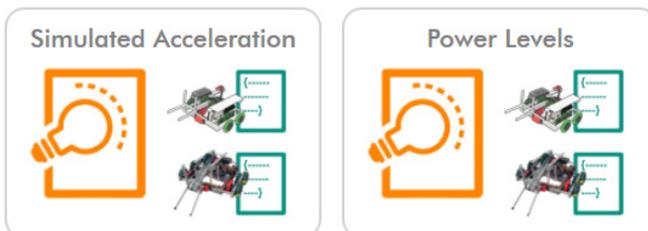
- b. Reversing Motor Polarity - This lesson shows the student how to program their robot to move backward by changing the polarity of their motors. The lesson set includes: an instructional video, RBC files, and Check Your Understanding questions.
- c. Renaming Motors - This lesson teaches students how to use the Motors and Sensors setup window to give custom names to individual robot motors. The lesson set includes: an instructional video, RBC files, and Check Your Understanding questions.
- d. Timing - This lesson teaches students how to adjust how long a motor is turned on and off using timing. The lesson set includes: an instructional video, RBC files, and Check Your Understanding questions.



Moving Forward Programming and Engineering Investigation - The Moving Forward lesson includes two simple programming and engineering investigations. The SumoBot Challenge requires student to change power levels and to investigate what that does to rotational torque. The Wait State Investigation requires students to change the amount of time that the robot moves and to record the distance that it travels. Both of these investigations can be completed in either a physical or virtual environment.

3. Speed and Direction Unit/Movement Chapter - 5 days

- a. Motor Power Levels - This lesson explains how to change the power levels programmatically. The lesson includes: an instructional video, RBC files, Check Your Understanding questions, and two programming investigations: Simulated Acceleration and Power Levels.



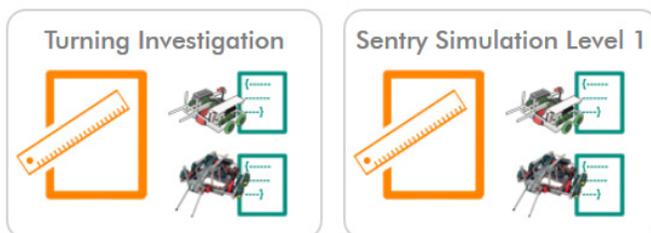
Continued next page

Chapter Overview Continued

Speed and Direction Unit/Movement Chapter Continued

The Motor Power Levels Unit includes to programming challenges. Simulated Acceleration is a very simple excise that requires students to simulate a robot accelerating. At this point students are still learning code and this is providing them practice writing code and understanding how pseudocode works. The Power Levels Engineering Investigation provides students with an opportunity to find out if there is a proportional relationship between Power Levels and distance travelled.

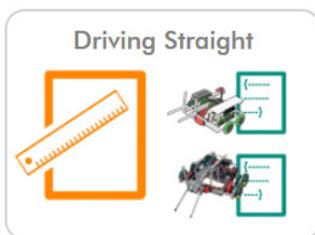
- b. Turn and Reverse - This lesson is designed to show students how to make different types of turns. The lesson includes an instructional video, RBC files, Check Your Understanding questions, and two programming investigations: the Turning Investigation and the Sentry Simulation Level 1.



The Turning Investigation is designed to provide students with the opportunity to practice with swing turns and point turns. They are asked to conduct an experiment that involves programming their robot to turn and then change a value and predict how far the robot will turn given the change in the value.

The Sentry Simulation requires students to program a robot using timing to march around a square. Students are required to answer questions about what happens when they change values in their programs.

- a. Manual Straightening - This lesson requires students to make adjustments to their robot to make sure that it goes perfectly straight. This is the most primitive way to program their robot and will not be used once they begin using sensor feedback. The lesson includes a video, RBC files, Check Your Understanding questions, and an additional programming challenge named: Drive Straight.



Continued next page

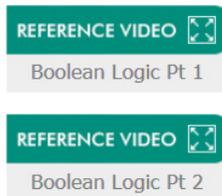
Chapter Overview Continued

3. Shaft Encoders - 5 days

- a. Shaft Encoders - This lesson explains how shaft encoders work. The lesson includes: an instructional video, RBC files, Check Your Understanding questions, and a PDF reference guide that explains how the shaft encoders work.
- b. Forward for Distance Part 1 - Explains how to program shaft encoders and how to use the Motors and Sensors Setup window. The lesson includes: an instructional video, RBC files, Check Your Understanding questions.
- c. Forward for Distance Part 2 - Continues to explain how to use and program shaft encoders and why it is important to clear the encoders and how a while loop works. This lesson includes an instructional video, Check for Understanding questions, RBC files, a reference PDF that explains how While Loops work, and two reference videos that explain Boolean Logic.

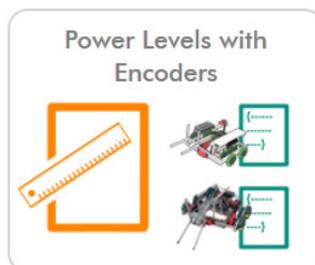


While Loop reference handout - this handout explains how while loops are used in programming.



Boolean Logic reference videos - Boolean Logic is used to help the robot to make decisions. It is critical that students understand this concept and is sometimes difficult for students to understand. These videos will be found in multiple places in the curriculum.

- d. The Sensor Debug Window - The debugger is an incredible tool that is included within ROBOTC. **MAKE SURE YOUR STUDENTS KNOW HOW TO USE THE DEBUGGER!** This lesson includes an instructional video, RBC files, Check Your Understanding questions, and two programming challenges: Basketball Drills and Power Levels with Encoders.



- e. Forward and Turning - This lesson completes the introduction to shaft encoders lesson and includes a summary video, Check Your Understanding questions, RBC files, and an additional Engineer Investigation called Turning with Encoders.

Continued next page

Chapter Overview Continued

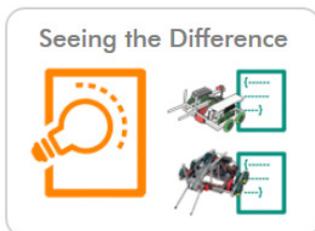
4. Automated Straightening Unit/Movement Chapter - 5 days

- a. Automated Straightening Part 1 - This lesson teaches students to use a combination of encoders and conditional statements to have the robot self correct its forward movements. The lesson includes an instructional video, RBC files, Check Your Understanding questions, and an “if-else” reference guide.



The if-else reference guide provides multiple examples with comments on how the if-else structure can be used.

- b. Automated Straightening Part 2 - Completes the video instruction. The lesson includes an instructional video, Check Your Understanding questions, RBC files, and two programming challenges: Driving Straight 2 and Seeing the Difference.



Driving Straight 2 challenges students to solve the same course that they solved earlier, but this time they are using feedback from sensors and an algorithm that they just developed. Hopefully they see that using sensors and programming is much better than motors and timing.

Seeing the Difference requires students to use the built in debugger. The debugger is a programmer’s best friend and this investigation is designed to give students practice using the debugger.

- c. Values and Variables Part 1- This teaches students about variables and the power of using variables when writing code. This lesson includes an instructional video, RBC files, Check Your Understanding questions, and two reference PDFs: Variables and Global Variables.



Variables



Global Variables

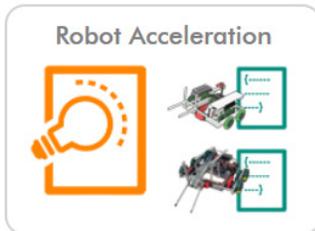
Variables are very important in programming. These two reference guides teach students about variable types, how to declare variables, and provide commented examples of variables being used in code.

Continued next page

Chapter Overview Continued

Automated Straightening Unit/Movement Chapter Continued

- d. Values and Variables Part 2 - This lesson is part two of values and variables. Students are required to create two variable in their program and use them to make programming more efficient. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, and the Robot Acceleration programming challenge.



Students completed this challenge several units ago. This time, they are able to create a variable and manipulate it mathematically.

5. Integrated Encoders Unit/Movement Chapter - 5 days

- a. Forward for Distance IME - This lesson teaches students how to use Integrated Motor Encoders. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, and an Integrated Motor Module reference guide produced by VEX Robotics.
- b. Principles of PID - This lesson teaches students about how a Proportional Integral Derivative controller works. This lesson set includes an instructional video and Check Your Understanding questions.
- c. Forward for Distance PID - Teaches students how to control the distance that the robot moves using IMEs and PID. The lesson set includes an instructional video, Check Your Understanding questions, and RBC files.
- d. Forward for Target Distance - This lesson teaches students how to stop in an exact location using the MoveMotorTarget command. They will also learn about idle loops and logical operators. They will need to review logical operators using the Boolean Logic 2 reference video. This lesson set includes the instructional video, RBC files, Check Your Understanding questions, and two programming challenges: Basketball Drills and Sentry Simulation.



Students completed this challenge several units ago using only timing, now they can use motor encoders and PID and they should see that they can program their robot to move more accurately this time.

Continued next page

Chapter Overview Continued

Chapter: Remote Control - 15 days

1. The Minefield Challenge/Remote Control Chapter

- a. The Minefield Challenge - this video and PDF show students what the culminating activity for the chapter will look like. In this chapter, students learn how to program their VEXnet Joystick.

Pseudocode and Flowcharts - 3 days then ongoing

Note: It is important to require students to use pseudocode and flowcharts now! Students will develop their initial code using pseudocode and then develop a flowchart that shows the robot's decision making.

Find lesson resources on pages 17 - 25 in this teacher's guide.

2. Joystick Mapping Unit/Remote Control Chapter - 3 days

- a. Introduction to Remote Control - this video shows how remote control works with the VEX Cortex system. This lesson set includes a video, Check Your Understanding questions, and a VEXnet Joystick Calibration PDF.
- b. Real-Time Control - In this lesson students experiment with an existing sample program and attempt to optimize their real-time remote control. They will learn how the while loop works and Boolean Logic. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, and two reference handouts: While Loops, and Boolean Logic.



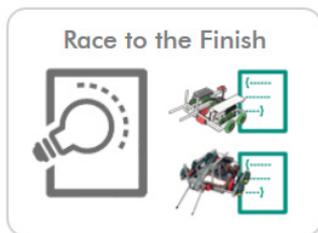
While()
Loops



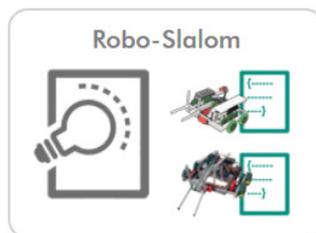
Boolean Logic

Students will use While Loops and Boolean Logic for most programming activities moving forward. These are important reference guides for them.

- c. Mapping Values Part 1 - This lesson set teaches students which joystick button and stick maps to which values.
- d. Mapping Values Part 2 - This video completes the Joystick Mapping lesson and teaches students how to adjust the speed of the motors. The lesson includes: instructional videos, RBC files, Check Your Understanding questions, and several programming challenges: Robo-Slalom and Race to the Finish.



Race to the Finish



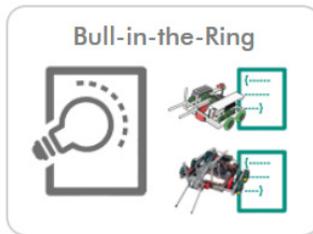
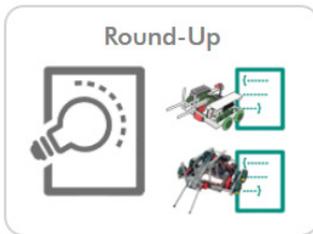
Robo-Slalom

Continued next page

Chapter Overview Continued

3. Timers Unit/Remote Control Chapter - 2 days

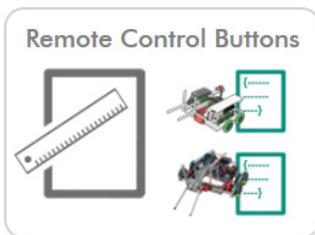
- Time and Timers- In the first lesson in this lesson set, students learn about Timers and how they can be used in programming. This lesson includes an instructional video, Check Your Understanding questions, and a PDF reference guide on using Timers with ROBOTC.
- Using Timers - In this lesson students learn to implement timers in their programs. This lesson includes an instructional video, RBC file, Check Your Understanding questions, and two programming challenges: Round Up, and Bull In the Ring.



Round Up and Bull In the Ring are two simple remote control programming challenges. Encourage students to modify the challenges to make them more engaging.

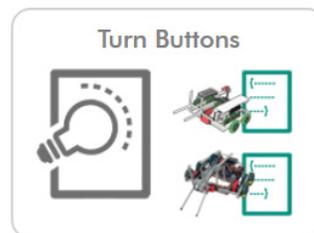
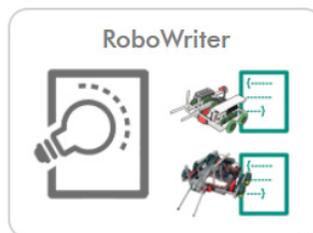
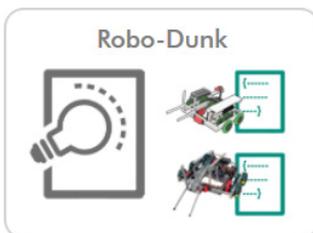
4. Buttons Unit/Remote Control Chapter - 5 days

- Remote Control Buttons - in this video lesson students learn how to identify and program the buttons on the top and front of the remote control.
- Remote Start - In this lesson students will learn how to program the robot to start using a button on the remote control. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, and a programming challenge named: Remote Control Buttons.



The Remote Control Buttons challenge is designed to guide students as they program the buttons on their VEX remote control. The programmable remote control is a very powerful tool in robotics competitions. Encourage students to program robot behaviors to their buttons. For example, BtnX makes the robot turn right.

- Controlling the Arm Part 1, Part 2, & Part 3 - This lesson provides a step-by-step set of instructions to program their robot's arm to be controlled by the remote control. The lesson set includes three instructional videos, RBC files, Check Your Understanding questions, and three practice programming challenges.



5. Conduct a Minefield Challenge In-class Competition! - 2 days

Continued next page

Chapter Overview Continued

Chapter: Sensing - 25 days

1. The Grand Challenge/Sensing Chapter

- The Grand Challenge is a teacher designed challenge that provides an opportunity for the student to demonstrate that they know all of the sensors, how to write functions and pass parameters, and can write reusable code. Tell students in advance by the end of the chapter that they will need to write functions that enable their robot to use feedback from sensors to solve multiple programmable tasks. The video and PDF on the Grand Challenge page show an example of what the challenge might look like.

2. Limiting the Arm Unit/Sensing Chapter - 5 days

- Configuring Sensors - this lesson teaches students how to use the Motors and Sensor Setup window to configure the robot's sensors. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, and two reference guides: Sensors and Servo Modules.



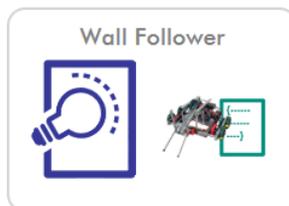
Sensors
(Inventor's Guide)



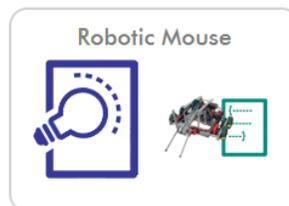
Servo Modules

The Sensors and the Servo Modules reference guides explain how VEX sensors and servo modules work.

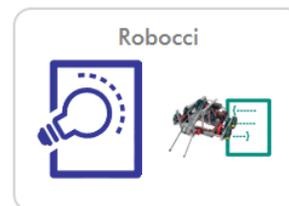
- Limiting the Arm Part 1 and Part 2 - In this lesson students will learn how to use both the touch sensor and the potentiometer to control the arm on the robot. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, several additional programming challenges and reference handouts.



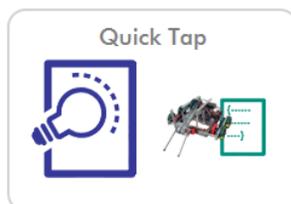
Wall Follower



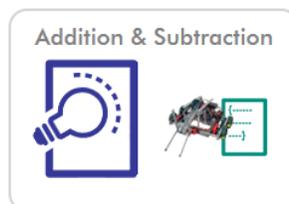
Robotic Mouse



Robocci



Quick Tap



Addition & Subtraction

These five challenges can only be completed using physical robots. The if-else, Switch Cases, and Potentiometer reference guides provide the student with sample code that helps them to understand how these structures work.



if-else
Statements



Switch Cases



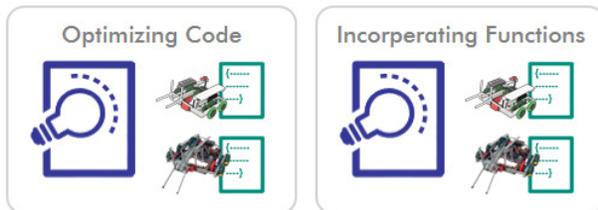
Potentiometers

Continued next page

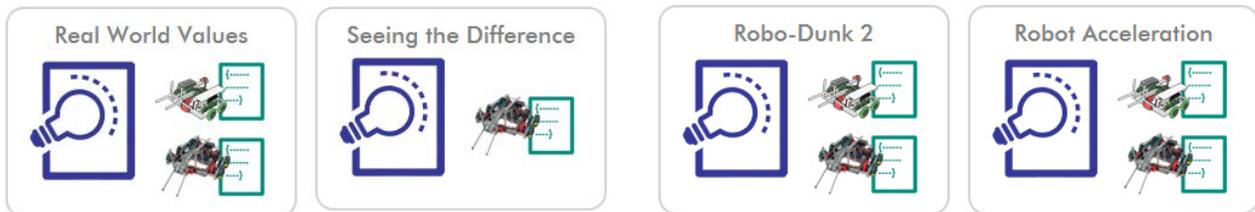
Chapter Overview Continued

3. Behaviors and Functions Unit/Sensing Chapter - 7-8 days

- a. Behaviors and Functions Part 1 and Part 2 - In this lesson set students will learn how functions can make their programs shorter and easier to follow. The lesson set includes: two instructional videos, RBC files, Check Your Understanding questions, and two programming challenges.



- b. Passing Parameters Part 1 and Part 2 - In this lesson set students learn about the power of using parameters in functions. The lesson set includes: two instructional videos, RBC files, Check Your Understanding questions, and four programming challenges.



Seeing the difference can only be completed using a physical robot.

4. Forward Until Near Unit/Sensing Chapter - 5 days

- a. The Ultrasonic Rangefinder - In this lesson students will learn what an Ultrasonic Rangefinder is and how it works. The lesson includes: an instructional video, Check Your Understanding questions, and the Ultrasonic Rangefinder reference guide.



- b. Forward until Near - In this lesson student will learn how to write a program that uses the ultrasonic rangefinder. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, several additional programming challenges and a Thresholds reference handout.

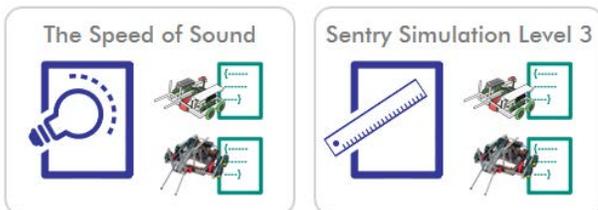


Continued next page

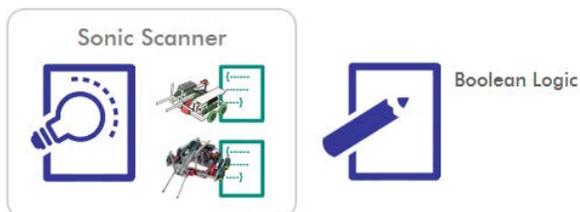
Chapter Overview Continued

4. Forward Until Near Unit/Sensing Chapter Continued

- d. Straight Until Near - In this lesson students will learn how to implement the straight until near behavior. The lesson includes: an instructional video, RBC files, Check Your Understanding questions, and two programming challenges.



- e. Straight Until Near (Fine Tuning) - In this lesson student will complete their Forward Until Near coding. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, an additional programming challenge and the Boolean Logic reference guide.



Note: Logical operators can be tricky for students to understand. This may be a good time to review this concept with your class.

5. Line Tracking Sensors Unit/Sensing Chapter - 10 days

- a. Line Tracking Sensors - this lesson reviews the sensors students have used up to this point in the curriculum and then teaches students how VEX Cortex Line Tracking sensors work. This lesson set includes an instructional video, RBC files, Check Your Understanding questions, and two reference guides: Build instructions for Swervebot and a Line Follower reference guide.



Note: Linetracking is difficult with some robot types. You may want to have a specially built robot specifically for this lesson because it takes a very long time for students to take apart and build a new robot. Another option is to complete the lesson using the Robot Virtual World simulation software.

Continued next page

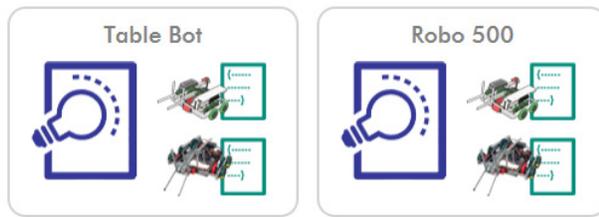
Chapter Overview Continued

Line Tracking Sensors Unit/Sensing Chapter Continued

- b. Calculating Thresholds - In this lesson students will learn how to calculate the a threshold value using feedback from the ROBOTC debugger. The lesson includes: an instructional video, RBC files, Check Your Understanding questions, a programming challenge, and two reference guides.



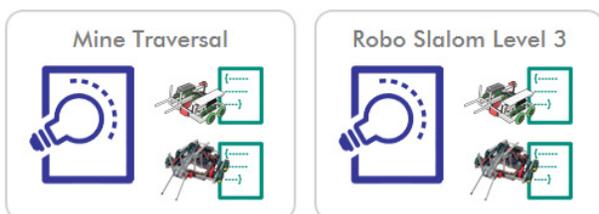
- c. Basic Line Tracking - This lesson teaches students how to write a program that will complete a basic line tracking behavior. The lesson includes an instructional video, RBC files, Check Your Understanding questions, and two programming challenges.



- d. Line Tracking for Distance - In this lesson students will learn how to use feedback from multiple sensors to track a line and stop at a specific distance. The lesson includes: an instructional video, RBC files, Check Your Understanding questions, a programming challenge.



- e. Optimized Line Tracking - In this lesson students will learn how to optimize their motor speeds in order to track lines more efficiently. The lesson includes: an instructional video, RBC files, Check Your Understanding questions, and two programming challenges.

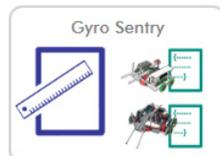
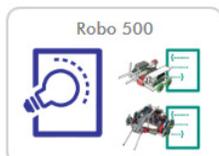


Continued next page

Chapter Overview Continued

6. Turn for Angle Unit/Sensing Chapter - 5 days

- The Gyro Sensor - In this lesson students will learn how the Gyro Sensor works. The lesson includes: an instructional video and Check Your Understanding questions.
- Turn for Angle Part 1 and Part 2 - In this lesson set students will learn how to program their robot to use feedback from a gyro sensor to perform a measured turn. This lesson set includes two instructional videos, RBC files, Check Your Understanding questions, and three programming challenges.



Note: Students have seen variants of these robot challenges before. The difference now is that they have more programming tools to work with. Encourage students to modify the programming challenges to make them more interesting.

7. Intro to the LCD - 5 days

- The VEX LCD - In this lesson students will learn about the many uses of the VEX LCD. This lesson set includes an instructional video, Check Your Understanding questions, and the VEX LCD reference guide.



- Displaying Text - In this lesson teaches students how to use the VEX LCD as an output device. The lesson includes an instructional video, RBC files, and Check Your Understanding questions.
- Displaying Sensor Values - In this lesson teaches students how to continually update the values on the LCD and to display current robot sensor values. The lesson includes an instructional video, RBC files, and Check Your Understanding questions.

Note: Have students go back to several previous programming challenges and to modify the programming challenges so that students can see real time sensing feedback on the LCD.

8. Compete in the Grand Challenge Competition - 5 days

The Grand Challenge is a teacher designed programming competition where the students don't get the actual challenge until two days before the competition. One week before the competition they will learn about the behaviors that their robot need to perform. For example, you robot will need to be able to complete the following behaviors: Stop at a black line, turn accurately using feedback from a gyro sensor, identify the distance from an object using feedback from an ultrasonic sensor, and identify the location of an object and pick it up and come home.

Continued next page

Chapter Overview Continued

Chapter: Engineering - Semester

1. Safety Unit/Engineering Chapter

- a. Safety - The safety reference guides and tests are to be used at the beginning of the semester to establish a safe work environment. The safety resources include the following handouts:
 - Safety is an Attitude
 - Safety (from the Inventor's Guide)
 - General Lab Safety
 - Electrical Safety
 - Power Tool Safety
 - The Safety Checklist
- b. Safety Tests - It is important to administer a safety test to establish the importance of safety in your classroom. The safety resource unit includes the following safety tests:
 - Safety Aptitude Test
 - General Safety Test
 - Safety Quiz
 - Robotics Lab Inspection Sheet

2. VEX Hardware Guides

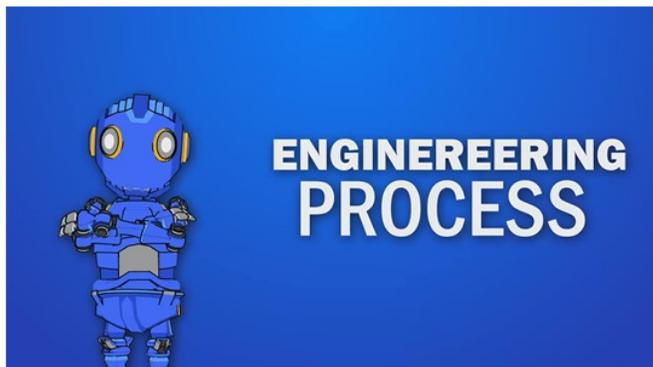
- a. Motors and Building Materials - This section includes great resources to teach new builders about Structural systems and how to build with VEX hardware. The resources include the following handouts:
 - Motion - a 26 page guide that shows how VEX motion systems are designed. This guide uses a large amount of pictures that show how parts connect.
 - Structure - a 17 page guide that shows new builders how parts go together and important considerations about weight distribution and balance.
 - Motor Handouts - VEX has multiple motor types, these handouts describe the technical use and capability of the various motors.
- b. Sensors and Displays - This section includes technical guides for the following VEX hardware:
 - Shaft Encoders
 - Potentiometers
 - Ultrasonic Rangfinder
 - Light Sensors
 - VEX LCD

Continued next page

Chapter Overview Continued

3. Engineering Process Unit/Engineering Chapter - Ongoing

- a. Engineering Process - This lesson is a foundational lesson designed to help students to think about engineering process generally. This lesson includes an instructional video and key handouts that describe process that students will use throughout the course.



The Engineering Process video at the left uses a combination of levity and graphics to teach students about how to solve and engineering challenge.

Keeping an Engineering Journal is a guide for students on what they should keep in the journal.



Engineering
Process
Reference



Keeping an
Engineering
Journal



Design Reviews



Understanding the
Problem



Brainstorming



Engineering
Definitions

- b. Project Planning - In this instructional video students learn the importance of project planning. The handouts provide insight on how to build a team, plan time, organize ideas, record project progress, and prepare for a competition.



Team Building



Gantt Chart



PERT Chart



Organizational
Matrix Ideas



Recording Progress



Preparation for a
Competition



Planning
Your Time

Continued next page

Chapter Overview Continued

3. Engineering Process Unit/Engineering Chapter Continued

- c. Engineering Automated Workcell - There are many engineering projects that your students can solve using VEX Robotics parts. This video shows an automated workcell made out of VEX robot parts.

4. Competition Programming

- a. Competition Programming Part 1 and Part 2 - Robot Competitions have strict rules about how programs should be written to compete in the competition. This lesson includes two instructional videos, RBC files, Check Your Understanding questions, and a step by step guide.



- b. The VEXnet Competition Switch - When you get to the competition, you will be using a VEXnet Competition Switch, not all schools have one. This instructional video and Check Your Understanding questions are designed to prepare you to use the VEXnet competition switch.
- c. The Programming Hardware Kit - This lesson will show you how you can test your competition program using the Programming Hardware kit.

5. Rubrics

- a. Assessment rubrics are common tools used for project based courses. These rubrics may not fit everything that you are doing in your classroom, but can be used as a guide to develop your own rubrics, or can be used as they are.



Writing Criteria



Workplace Competencies



Robotics Exploration Rubrics



Presentation Rubrics



Proposal Assessment Rubric



Internal Design Reviews



Working Habits



Engineering Journal Rubric



External Design Reviews

Navigating the Curriculum

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



The **Fundamentals and Setup** chapters provide resources that will be helpful to new VEX/ROBOTC students.

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

Fundamentals/Introduction to Programming

Robots are designed to solve specific problems, in specific ways. In this section, you will learn more about your role as the Programmer, along with some guidelines and tips when using ROBOTC.

COLLAPSE ALL

1. Introduction to Programming

Every journey needs a beginning. Click on one of the links below to begin your first lesson in robotics!

- 1. Programmer and Machine
- 2. Planning and Behaviors
- 3. ROBOTC Rules Part 1
- 4. ROBOTC Rules Part 2

2. Natural Language Programming

OPTIONAL: The ROBOTC Natural Language is designed to lower the barrier of entry into syntax-based programming by providing a set of robot behaviors that are easy-to-use and easy-to-remember.

- 1. Programming Reference Documents

The Fundamentals Unit is divided into two Lesson Sets: Introduction to Programming and Natural Language Programming

Fundamentals/Introduction to Programming

HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING ENGINEERING REFERENCE

1. Introduction to Programming 1 2 3 4

Programmer and Machine

FUNDAMENTALS
Programmer and Machine

© Copyright 2015 Carnegie Mellon Robotics Academy

Check Your Understanding:

- Q 1. A program is**
- A logical and step-by-step set of directions for the robot to follow
 - A logical and step-by-step set of directions for the programmer to follow
 - A set of directions, written in paragraph form, that even a robot can follow
 - None of the above
- Q 2. The role of the robot is to carry out the plan by following the steps in the program.**
- True
 - False
- Q 3. What is a programming language?**
- Only ROBOTC
 - A method for the robot to communicate with the programmer
 - A program that allows the robot to think for itself
 - A language that lets the programmer communicate a plan to the robot

PREV NEXT Next Lesson: Planning and Behaviors

VEX Cortex Video Trainer using ROBOTC
Copyright(c)2014 Carnegie Mellon Robotics Academy. All rights reserved.

The Programmer and the Machine

The “Programmer and Machine” video explains 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 video is designed to explain programming concepts to beginners.

Each page video includes a set of “check your understanding” questions that cover the main topics covered in the video.

Fundamentals/Introduction to Programming

HOME
FUNDAMENTALS
SETUP
MOVEMENT
REMOTE CONTROL
SENSING
ENGINEERING
REFERENCE

1. Introduction to Programming 1 2 3 4

Planning and Behaviors

FUNDAMENTALS
Planning and Behaviors

© Copyright 2015 Carnegie Mellon Robotics Academy

Check Your Understanding:

Q 1. Pseudocode is:

- False code
- A code, written in English, that the robot can understand
- A code, written in binary language, that the robot can understand
- A set of basic steps that the human can use to write the program

Q 2. _____ are a convenient way to talk about what the robot is doing and what it must do to carry out the plan created by the programmer.

- Operations
- Actions
- Behaviors
- Instructions

Q 3. Simple behaviors are made up of complex behaviors.

True

Behaviors

Pseudocode & Flowcharts

Thinking about Programming

REFERENCE VIDEO

Iterative Design

REFERENCE VIDEO

Flow Charts

Prev Lesson: Programmer and Machine ◀ ▶ Next Lesson: ROBOTC Rules Part 1

VEX Cortex Video Trainer using ROBOTC
Copyright(c)2014 Carnegie Mellon Robotics Academy. All rights reserved.

The Planning and Behaviors Video

The Planning and Behaviors introductory video explains behavior based programming to students. The lesson also includes: check your understandings questions, three handouts, and two additional videos which are shown below.

Reference

Behaviors

A behavior is something your robot does (turning on a single motor or a behavior, moving forward as a behavior, tracking a line is a behavior, navigating a maze is a behavior). There are three main types of behaviors that we are concerned with: basic behaviors, simple behaviors, and complex behaviors.

Basic Behaviors
Example: Turn on Motor Port 3 at half power.
All the most basic level, everything in a program must be broken down into tiny behaviors that your robot can understand and perform directly in ROBOTC. These are behaviors the size of single statements, like turning on a single motor, or resetting a laser.

Simple Behaviors
Example: Hit.
Simple behaviors that often are significant to the robot and useful in ROBOTC code.

Complex Behaviors
Example: Turn on Motor Port 3 at half power.
These are behaviors that are composed of down into one or more simple behaviors.

Reference

Pseudocode & Flow Charts

Flow Charts are a visual representation of program flow. A flow chart normally uses a combination of blocks and arrows to represent actions and sequences. Blocks typically represent actions. The order in which blocks occur is shown using arrows that point from block to block. Sometimes a block will have multiple arrows coming out of it, representing a way where a decision must be made about which path to follow.

Start and End symbols are represented as rounded rectangles, usually containing the word "Start" or "End", but can be more specific, such as "Start Motor On" or "Stop all motors".

Actions are represented as rectangles and are basic statements. Examples: "Set Motor(DD);", "Increment LineCount by 1;" or "Motor Off always".

Decision blocks are represented as diamonds. These typically contain "Yes/No" questions. Decision blocks have two or more arrows coming out of them, representing the different paths that can be followed, depending on the outcome of the decision. The arrows should always be labeled accordingly.

To the right in the flow chart of a program.

Fundamentals

Thinking About Programming Programmer & Robot

In this lesson, you will learn about the roles of the programmer and the robot, and how the two need to work together in order to accomplish their goals.

Robots are made to perform useful tasks. Each one is designed to solve a specific problem, in a specific way.

Autonomous Tractor
This tractor through a field, it can detect the location of the tractor and the location of the field.

Autonomous Robot
This robot is designed to move in a straight line.

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.

ITERATIVE DESIGN

HOW DO ROBOTS THINK?

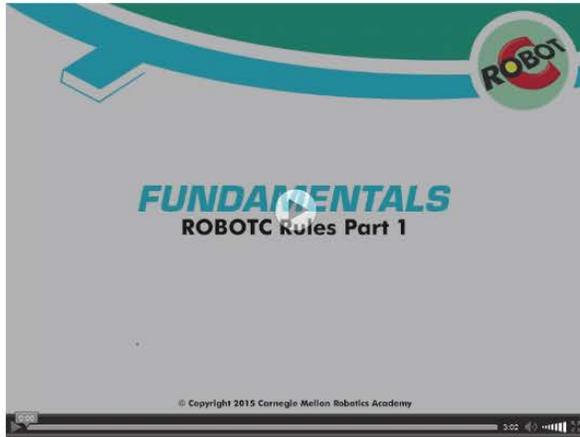
How Do Robots Think and Iterative Design - These two new videos are designed to help students understand how to break their programs into simple logic and behaviors.

Fundamentals/Introduction to Programming



1. Introduction to Programming 1 2 3 4

ROBOTC Rules Part 1



Check Your Understanding:

Q 1. Which one of these representations of Task Main will work in ROBOTC?

- Task main ()
- task main ()
- Taskmain ()
- task main;
- all of them

Q 2. What does it mean when a command appears in color when typed in ROBOTC?

- The command is typed incorrectly
- The command is not allowed in the command
- ROBOTC recognizes the command as an important keyword
- It is highlighted so the programmer should look at the word or name carefully

Q 3. All programs must end with a 'Stop' command.

- True
- False

Q 4. The purpose of 'whitespace' is:

- To allow the robot to understand the program better
- To make the program more readable to the programmer
- To speed the operation of the robot



Whitespace

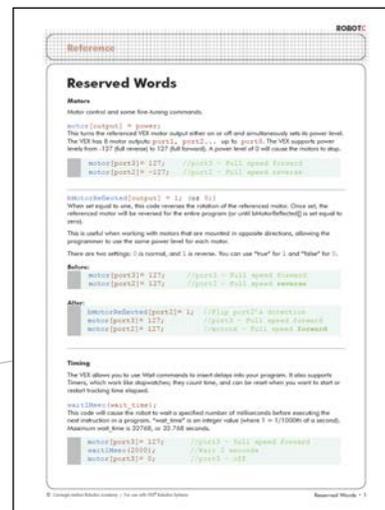
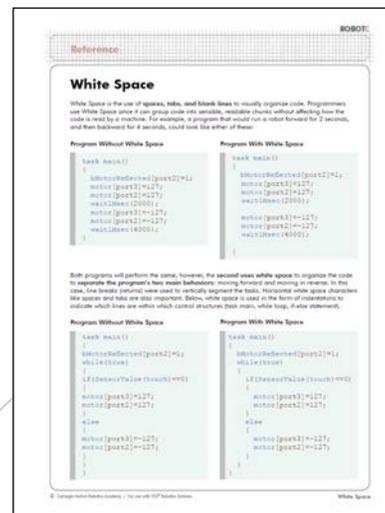


Reserved Words

Prev Lesson: Planning and Behaviors: [PREV](#) [NEXT](#) Next Lesson: ROBOTC Rules Part 2

VEX Cortex Video Trainer using ROBOTC
Copyright(c)2014 Carnegie Mellon Robotics Academy. All rights reserved.

The ROBOTC Rules Part One includes: The introductory video, check your understanding questions, and the two Whitespace and Reserved Words handouts shown below.



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

Fundamentals/Introduction to Programming

The screenshot shows the ROBOTC software interface. At the top, there is a navigation bar with icons for HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. Below this, a breadcrumb trail shows '1. Introduction to Programming' with sub-sections 1, 2, 3, and 4. The main content area features a video player titled 'ROBOTC Rules Part 2' with a play button and a 'FUNDAMENTALS' logo. The video player shows a title card for 'FUNDAMENTALS ROBOTC Rules Part 2' and a copyright notice for 2015 Carnegie Mellon Robotics Academy.

The **ROBOTC Rules Part Two** includes: The introductory video, check your understanding questions, and the three handouts shown below.

Check Your Understanding:

- Q 1. If one of a paired punctuation such as parentheses is missing:**
- The robot will assume that it is there and continue running
 - The program will not compile and the programmer will see an error message
 - The program will bypass that statement
 - The program will delete the paired punctuation that is not missing
- Q 2. Pick the true statement:**
- Paired punctuation are interchangeable. i.e., You can replace a pair of square brackets with a pair of curly braces
 - As long as there are two brackets of any kind in a pair, the program will work
 - Curly braces surround what the robot will run as part of the main program, or task
 - Motor command must use parentheses and wait for commands must use square brackets
- Q 3. A simple statement may affect the order that statements are run, allowing the program to skip or repeat commands as needed**
- True
 - False



Comments

Commenting a program means using descriptive text to explain portions of code. The compiler and robot will ignore comments when running the program, allowing a programmer to leave important notes in source code format, or to organize the program code itself. This is considered very good programming style, because it can help you avoid potential confusion later on when someone else (or even you) may need to read the code.

There are two ways to mark a section of text as a comment rather than normal code:

Type	Start Notation	End Notation
Single line	//	
Multiple line	/*	*/

Below is an example of a program with single and multi-line comments. Commented text turns green.

```

// This program uses commenting
// to describe each sensor.

task main()
{
  MicroSwitched[port2] = 1; //robot watches on port 2
  motor[port3]=127; //port3 clockwise Full speed
  motor[port2]=127; //port2 clockwise Full speed
  wait1Msec(1000);
}
    
```

Error Messages in ROBOTC Code

ROBOTC has a built-in compiler that analyzes your programs to identify syntax errors, capitalization and spelling mistakes, and code formatting issues to prevent accidents. The compiler runs every time you download code to the robot and when you choose to compile your program from the Robot menu in ROBOTC.

Notifications regarding any errors, warnings and important information the compiler finds are displayed on the Error display screen of the ROBOTC interface.

The Error display screen reports the number of errors in your code, as well as their types. Clicking a compiler message in the Error display screen will highlight the relevant line of code in your program. Depending on the type of error, ROBOTC will only be able to highlight the appropriate location. For instance, in the example above the missing semicolon is on line 7 but ROBOTC will highlight line 6.

ROBOTC generates three types of compiler messages: **Errors**, **Warnings** and **Information**.

- Errors:** There was an issue your program that prevented it from compiling. These are usually misspelled words, missing semicolons, and improper syntax. Errors are denoted with a **Red X**.
- Warnings:** There was a minor issue with your program, but the compiler was able to fix or ignore it. These are usually incorrect capitalizations or empty, infinite loops. Warnings are denoted with a **Yellow W**.
- Information:** ROBOTC will generate information messages when it thinks you have declared functions or variables that are not used in your program. These messages inform you about inefficient programming. Information messages are denoted with a **Grey I**.

ROBOTC is a task-based programming language

Commands to the robot are first written as text on the screen. They are then processed by the ROBOTC compiler into a machine language for the robot to understand. Finally, they are loaded onto the robot, where they can be run.

```

task main()
{
  motor[port2]=127;
  wait1Msec(1000);
}
    
```

For written as part of a program to called code. This text code can be like you type normal text. You do not need that capitalization is programmed to the computer. Replacing a lower case letter with a capital letter or a capital letter with a lower case letter will cause the robot to become confused.

```

task main()
{
  motor[port2]=127;
  wait1Msec(1000);
}
    
```

Capitalization: Capitalization (using uppercase) is important in ROBOTC. For example the "T" in task, ROBOTC, or motor are important.

As you type, ROBOTC will try to help you not by coloring the words it recognizes. If a word appears in a different color, it means ROBOTC recognizes it as an important word in the programming language.

```

task main()
{
  motor[port2]=127;
  wait1Msec(1000);
}
    
```

Code Syntax: ROBOTC automatically adds the words task and program. Change the word "task" to "task" and the word "main" to "main". The word "main" is required as a command to run from the robot.

Comments, ROBOTC Error Messages, and ROBOTC Rules - These handouts are designed to complement the ROBOTC Rules programming videos above.

Fundamentals/Natural Language Programming

The screenshot shows the ROBOTC software interface. At the top, there is a navigation bar with tabs: HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. Below the navigation bar, the '2. Natural Language Programming' section is active, indicated by a blue bar and a '1' in a square. The main content area is titled 'Reference Documents' and contains a grid of 24 document icons, each with a blue pencil icon and a title. The titles are: Vex Cortex Reference, VEX PIC Reference, Sense, Plan, Act, Behavior Based Programming, Pseudocode & Flowcharts, Comments, Boolean Logic, White Space, Thresholds, Variables, While Loops, if-else Statements, Functions, Timers, Shaft Encoders, Ultrasonic Rangefinder, Touch Sensors, Potentiometers, Line Follower, Light Sensor, Accelerometer, VEX LED, Servo Motors, VEX Claw, and VEX Flashlight.

What is Natural Language?

Natural Language provides an intuitive, easy to use, English language version of ROBOTC. Natural Language is for beginner programmers and is designed as a stepping stone to full ROBOTC programming. The Natural Language Library is filled with commands that are both easy to use and easy to remember. Natural Language commands encompass entire robot behaviors into a single command. The documents provided with the curriculum were current with the date the curriculum was developed. If you are looking for the latest features of ROBOTC's Natural Language consult the ROBOTC Wiki at: http://www.robotc.net/wiki/VEX2_Natural_Language.

This thumbnail shows the first page of the 'ROBOTC Natural Language - VEX PIC Reference' document. It includes a 'Setup Functions' section with a 'Robot Type' parameter, a 'Valid Robot Types for type' section, and a 'Move Forward Functions' section with a 'Set Servo' parameter. The document is formatted with clear headings and code examples.

This thumbnail shows the first page of the 'ROBOTC Natural Language - VEX Cortex Reference' document. It includes a 'Setup Functions' section with a 'Robot Type' parameter, a 'Valid Robot Types for type' section, and a 'Move Forward Functions' section with a 'Set Servo' parameter. The document is formatted with clear headings and code examples.

VEX PIC and VEX Cortex Natural Language reference guides. These two PDFs contain examples of the code and explanations of how the code works.

Note: Natural Language is a good choice if you have a very short amount of time to dedicate to teaching programming. BUT, when students begin to write complex programs Natural Language has limitations. If you are teaching a semester or a full year course it is recommended that you begin by teaching traditional ROBOTC.

Fundamentals/Natural Language Programming - Additional handouts that explain ROBOTC NL programming. The handouts can be used a study guides or quick reference sheets. They are available online so that they can be used for homework.

Reference

Ultrasonic Rangefinder Overview



The Ultrasonic Rangefinder, pictured left, is used to detect objects at a distance, without the need for the robot to actually contact them. The Ultrasonic Rangefinder uses sound waves to measure distance, in a similar way to how bats use echolocation. By emitting a sound wave and timing how long it takes to bounce back, the Ultrasonic Rangefinder can accurately estimate how far away the object is.



This sensor is most often used to detect obstacles. It can detect objects at a range of up to 200 inches (approximately 508 centimeters), but this range can be limited in certain situations. Since the Ultrasonic Rangefinder uses sound waves, any surface that absorbs or reflects sound, such as cardboard surfaces or shiny objects.

When using the Ultrasonic Rangefinder in the Cortex, both sensors are plugged into adjacent digital sensor ports. For the sensor to work properly, the "TRIP" wire must be in the higher numbered slot, and the "OUTPUT" wire in the lower numbered slot. For example, one wiring configuration is having the "TRIP" wire in digital port 6, and the "OUTPUT" wire in digital port 5.

Note: The units that an Ultrasonic Rangefinder uses (inches, centimeters, etc.) are controlled by the user in the "Sensors and Sensors Setup" window.

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Ultrasonic Rangefinder 1

Reference

Light Sensor Overview



The VEX Light Sensor allows the robot to sense the amount of light in the area. Unlike the Ultrasonic Rangefinder, the Light Sensor does not generate any sound waves. The amount of light already present in an area.



The VEX Light Sensor is an analog sensor, and it returns values in the range of 0 to 255. Higher values indicate that the sensor is in a darker area, and lower values indicate lighter areas.

Some uses of the Light Sensor would be to have your robot stop moving whenever the lights in a room are switched off, or to sense when it moves into a shadow.

When attaching the Light Sensor to your Robot, remember that it needs constant light, so that it functions best when facing out into an open space.

The Light Sensor can reliably sense light from up to at least 100 feet from a light source. The sensor only senses visible light, and is not sensitive to infrared light.

Note: Remember that it is important to calibrate your Light Sensor by calculating a threshold value. For more help on calculating thresholds, see the "Thresholds with Natural Language" handout.

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Light Sensor 1

Reference

Claw Kit Overview and Natural Language Sample Code



The VEX Claw Kit is a full size claw robot that is fully compatible with other VEX parts. The claw itself consists of two motorized grippers joined together using only a handful of screws.

The claw, pictured left, comes partially pre-assembled, but notably does not include a motor or sensor, you will need to order one separately. Since it uses a standard VEX motor, programming the claw is relatively simple.

VEX Claw Kit

Claw Using Motor
This code sample uses a motor in Port 7 to control the movements of the claw. The Claw will open for one second, pause for one second, and then close for one second. Make sure that the claw is in the "closed" position before running the program.

```

TASK MAIN()
{
  motor[port7] = 50; //Open the claw motor to open
  wait(1); //Wait for 1 second
  motor[port7] = 50; //Stop the claw motor
  wait(1); //Wait for 1 second
  motor[port7] = -50; //Close the claw motor to close
  wait(1); //Wait for 1 second
}
    
```

Claw Using Sensor
This code sample uses a sensor in Port 7 to control the movements of the claw. The Claw will open to position 100, pause for one second, and then close to position 20. Note that your sensor position for "Open" and "Closed" may be different depending on how your sensor is mounted.

```

TASK MAIN()
{
  setSensor[port7, 200]; //Open claw to position 100
  wait(1); //Wait for 1 second
  setSensor[port7, -20]; //Close claw to position 20
}
    
```

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Claw Kit 1

Reference

Pseudocode & Flow Charts

Pseudocode is a shorthand notation for programming which uses a combination of informal programming structure and natural descriptions of code. Pseudocode is placed on representing the behavior or outcome of each portion of code rather than on strictly correct syntax if does not need to be reproducible, though.

In general, pseudocode is used to outline a program before translating it into proper syntax. This helps in the initial planning of a program, by creating the logical framework and sequence of the code. An additional benefit is that because pseudocode does not need to use a specific syntax, it can be translated into different programming languages and is therefore somewhat universal. It captures the logic and flow of a solution without the bulk of actual syntax.

Below is some pseudocode written for a program which controls a motor and an LED as long as a touch sensor is not pressed. A motor turns on and an LED turns on if an object is detected within 20cm of a sensor sensor, the motor turns off and an LED turns on if an object is detected within 20 cm.

```

TASK MAIN()
{
  while ( touch sensor is not pressed )
  {
    (motor detects object < 20cm away)
    Right Motor runs Forward
    and LED turns off
  }
  else
  {
    Right Motor stops
    and LED turns on
  }
}
    
```

Note: Indentation is used to show sub-sections in the code. In this case, the code is indented to show that the motor and LED are controlled by the same condition. Indentation is used to show that the motor and LED are controlled by the same condition.

Restrictions: Pseudocode is not a programming language. It is a shorthand notation for programming which uses a combination of informal programming structure and natural descriptions of code. Pseudocode is placed on representing the behavior or outcome of each portion of code rather than on strictly correct syntax if does not need to be reproducible, though.

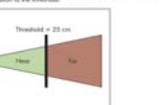
The pseudocode example includes elements of both programming languages, and the English language. Code blocks are used as a visual aid for where portions of code need to be placed when they are finally written into a real program.

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Pseudocode & Flow Charts 1

Reference

Thresholds with Natural Language

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 when a certain value is finally a sensor value falls relative to the threshold.



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

```

if (UltrasonicRangefinder[25])
{
  //Behavior
}
    
```

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

Calculated Thresholds
Some sensors, like the Ultrasonic Rangefinder and Inclinometer, provide the same set of values no matter what environment the robot is in, for the most part. Their thresholds can simply be chosen based on their application. Other sensors, like the Light Sensor and Line Tracking Sensors, will provide very different values based on the environment they're in. As a result, you'll need to calculate a threshold value based on the environment they're in. For example, to find a threshold for a light sensor, you would first calculate a threshold to distinguish light from dark. One recommended method:

1. Measure the Line Follower Sensor value of the light surface. (For more information on finding sensor values, reference the ROBOTC Debugger document.)
2. Measure the Line Follower 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: $\text{light value} + \text{dark value} / 2 = \text{threshold}$

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Thresholds 1

Reference

Functions with Natural Language

A function is a group of statements that are run as a single unit when the function is called from another location, such as in a 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 code:

- 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 more sub-behaviors).
- Through the use of parameters, multiple related (but not identical) tasks can be handled with a single, reusable function.

Using Functions
Functions must be created and then run separately. A function is created by "declaring" it, and then 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 might be give the function a name that reflects the behavior it will perform. Without the function's curly braces, write the commands exactly as you would normally. Remember that its name includes the parentheses - followed by a semicolon.
2. **Call Your Function**
Once you declare your function, it asks like a new command in the language of ROBOTC. To use the function, simply "call" it by name. Remember that its name includes the parentheses - followed by a semicolon.

```

void doSomething()
{
  //statements (statements);
}

TASK MAIN()
{
  doSomething();
}
    
```

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Functions 1

Reference

Touch Sensors Overview

There are two different VEX sensors, the Bumper Sensor and the Limit Switch, that act as touch sensors. Despite their differences in operation and usage, both sensors operate in a similar manner, and can be programmed in the same way.



Bumper Switch
The Bumper Switch returns a digital readout sensor for the VEX. Due to its size and construction, it is better suited to be used with a robot.



Limit Switch
The Limit Switch is a smaller form of touch sensor for VEX. It is more suited for attaching to a robot's frame to trigger the sensor sensor.



Both the Bumper Switch and the Limit Switch are digital sensors. Whenever the sensor is pressed in, it will return a value of 1. Whenever it is not pressed in, it will return a value of 0.

Touch sensors are used to perform a variety of tasks. From sensing a rising using wall detection to controlling the movements of your robot's arm attachment.

Note: On both the Bumper Switch and the Limit Switch, the red wire is not used. Even if you notice that the red wire is "floating out" of the sensor, it should not have any effect on the sensor's performance.

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Touch Sensors 1

Reference

Accelerometer



The VEX Accelerometer is an Analog Sensor that is able to measure acceleration on three axes (vertical, left/right, and forward/backward) at the same time. This data could allow the robot to calculate its velocity or even to calculate that it is not flat on a touch sensor. For comparison, there is an infrared light on the sensor itself that will light up solid green when the sensor detects that it is not properly. Additionally, the Accelerometer has connectors for connecting directly to other VEX construction parts, making it easy to integrate on almost any robot.



There are two sensitivity modes on the Accelerometer: a +2g mode when the sensor is not attached, and a +8g mode when the sensor is attached. The +8g mode gives more accurate readings, but the sensitivity mode will be more reliable for measurements that change quickly.

While the Accelerometer has the ability to take measurements in three axes of space, it is important to note that each axis needs its own Analog Trip, meaning that you could potentially use 3 Analog Ports per sensor.

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Accelerometer 1

Reference

Flashlight Overview and Natural Language Sample Code



The VEX Flashlight is an array of 4 LEDs that help you illuminate the way. The VEX Flashlight can plug into any motor port, but not all of them allow for brightness control. To have brightness control over the Flashlight, it needs to be either plugged into port 1 or port 10, or you need to use the "Motor Controller 2P" on ports 2 through 9. Without Motor Controller 2P, you can still use the Flashlight in ports 2 through 9, but you can only turn it either on the only on or completely off.



When plugged in to ports 1 or 10, the range of brightness values starts at 0 (off), and goes to 127 (full on).

When plugging a VEX Flashlight into ports 2 through 9, it is important to insert the wires into the two slots closest to the numbered part of the motor ports, not the side with the tab slots (see picture at bottom of page).

Also, when connecting the Flashlight to a Motor Controller 2P, be sure to make that the wires align properly. Red should connect to red, and black to black, as shown in the picture to the left.



Connect wiring for a VEX Flashlight to a motor port using a 10-pin header, make sure that it is plugged into the "Red" and "Black" slots.

© Copyright 2016 RobotC Academy. For use with VEX Robotics Systems. Flashlight 1

Fundamentals/Natural Language Programming - Additional handouts that explain ROBOTC NL programming. The handouts can be used as study guides or quick reference sheets. They are available online so that they can be used for homework.

Reference

Comments with Natural Language

Commenting a program means using descriptive text to explain portions of code. The compiler and robot both ignore comments when running the program, allowing a programmer to leave important notes in-line code through right alongside the program code itself. This is considered very good programming habit, because it adds an on-demand confusion later on when someone else for even just need need to read the code.

There are two ways to mark a section of text as a comment rather than normal code:

Type	Start Notation	End Notation
Single line	//	(none)
Multiple line	/*	*/

Below is an example of a program with single and multi-line comments. Commented text turns green.

```

// This is a portion of a multi-line comment.
// This program uses commenting to illustrate some features.
// This is the end of a multi-line comment.

task main()
{
  startMotor(ACTMOTOR1, 43); //Start motor1 on an 43 power
  wait1Sec(100); //Wait 100 seconds, which will be finished
  stopMotor(ACTMOTOR1); //Stop the motor1.
}

/*Commenting out Code
Commenting is also sometimes used to temporarily "hide" code in a program without actually deleting it. In the program below, the programmer has code to move and set up and then drive the arm down. However, in order to test only the second half of the program, the programmer made the first behavior only a comment, so the robot will ignore it. While the programmer is done testing the second behavior, he/she can remove the // comment marks to re-enable the first behavior in the program.
*/
task main()
{
  //startMotor(ACTMOTOR1, 43); //Start motor1 on an 43 power
  //wait1Sec(100); //Wait 100 seconds, which will be finished
  //stopMotor(ACTMOTOR1); //Stop the motor1.
  //Wait 1.5 seconds
  wait1Sec(150); //Wait 150 seconds
  startMotor(ACTMOTOR2, -43); //Start motor2 on an -43 power
  wait1Sec(100); //Wait 100 seconds
  stopMotor(ACTMOTOR2); //Stop the motor2.
}
    
```

© Copyright RobotC Systems | For use with ROBOTC Systems

Reference

Variables with Natural Language

Variables are places to store values (such as sensor readings) for later use, or for use in calculations. There are three main steps involved in using a variable:

1. Introduce (create or "declare") the variable
2. Give ("change") the variable a value
3. Use the variable to access the stored value

```

task main()
{
  //introduce
  int speed;
  //change
  speed = 75;
  //use
  startMotor(motor3, speed);
  startMotor(motor2, speed);
  wait1Sec(10);
}
    
```

Declaration
The variable is created by announcing its type, followed by its name, such as in an integer.

Assignment
The variable is assigned a value. The variable name is combined with the integer value 75.

The variable can now "store" the any value of the appropriate type, and will use it to drive other parts of the program.

Use
Each variable commands robot outputs for some getting on the it variable speed and drive it. The commands of their respective order priority to the value stored in speed, 75.

In the example above, the variable "speed" is used to store a number, and then retrieve and use that value when it is called for later on. Specifically, it stores a number given by the programmer, and retrieves it later on the two different places that it is used, once for each of the startMotor commands. This way both motors use set to the same value, but more interestingly, you would only need to change one line of code to change both motor powers.

```

task main()
{
  //introduce
  speed = 75;
  //use
  startMotor(motor3, speed);
  startMotor(motor2, speed);
  wait1Sec(10);
}
    
```

One line changed
The value changes to speed is now 50 instead of 75.

Changed without being changed
In the example above, the variable "speed" is used to store a number, and then retrieve and use that value when it is called for later on. Specifically, it stores a number given by the programmer, and retrieves it later on the two different places that it is used, once for each of the startMotor commands. This way both motors use set to the same value, but more interestingly, you would only need to change one line of code to change both motor powers.

This example shows just one way in which variables can be used, as a convenience for the programmer. With a robot, however, the ability to store sensor values (data that are measured by the robot, rather than set by the programmer) adds considerable new capabilities. It gives the robot the ability to take measurements in one place and deliver them in another, or even do some calculations on the stored values. The same basic rules are followed, but the possibilities go far beyond just what you've seen so far!

© Copyright RobotC Systems | For use with ROBOTC Systems

Reference

Timers

Timers are very useful for performing a more complex behavior for a certain period of time. Most robots built on VEX II don't set the robot outputs 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, a Timer must be used.

```

task main()
{
  robotTurn(robot02);
  clearTimer(T1);
  while (time1(T1) < 2000)
  {
    //do something
  }
  //do something else
  stop();
  printText("right, 150");
  wait1Sec();
  stop();
}
    
```

Clear the Timer
Changing the time count will give the time you set it to. You can choose to reset any of the timers. Here T1.

Time in the (parallel)
The time you set the timer, "clear" the timer, then you can choose to reset any of the timers. Here T1.

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

```

ClearTimer(T1);
ClearTimer(T2);
ClearTimer(T3);
ClearTimer(T4);
ClearTimer(T5);
ClearTimer(T6);
ClearTimer(T7);
ClearTimer(T8);
ClearTimer(T9);
ClearTimer(T10);
ClearTimer(T11);
ClearTimer(T12);
ClearTimer(T13);
ClearTimer(T14);
ClearTimer(T15);
ClearTimer(T16);
ClearTimer(T17);
ClearTimer(T18);
ClearTimer(T19);
ClearTimer(T20);
ClearTimer(T21);
ClearTimer(T22);
ClearTimer(T23);
ClearTimer(T24);
ClearTimer(T25);
ClearTimer(T26);
ClearTimer(T27);
ClearTimer(T28);
ClearTimer(T29);
ClearTimer(T30);
ClearTimer(T31);
ClearTimer(T32);
ClearTimer(T33);
ClearTimer(T34);
ClearTimer(T35);
ClearTimer(T36);
ClearTimer(T37);
ClearTimer(T38);
ClearTimer(T39);
ClearTimer(T40);
ClearTimer(T41);
ClearTimer(T42);
ClearTimer(T43);
ClearTimer(T44);
ClearTimer(T45);
ClearTimer(T46);
ClearTimer(T47);
ClearTimer(T48);
ClearTimer(T49);
ClearTimer(T50);
ClearTimer(T51);
ClearTimer(T52);
ClearTimer(T53);
ClearTimer(T54);
ClearTimer(T55);
ClearTimer(T56);
ClearTimer(T57);
ClearTimer(T58);
ClearTimer(T59);
ClearTimer(T60);
ClearTimer(T61);
ClearTimer(T62);
ClearTimer(T63);
ClearTimer(T64);
ClearTimer(T65);
ClearTimer(T66);
ClearTimer(T67);
ClearTimer(T68);
ClearTimer(T69);
ClearTimer(T70);
ClearTimer(T71);
ClearTimer(T72);
ClearTimer(T73);
ClearTimer(T74);
ClearTimer(T75);
ClearTimer(T76);
ClearTimer(T77);
ClearTimer(T78);
ClearTimer(T79);
ClearTimer(T80);
ClearTimer(T81);
ClearTimer(T82);
ClearTimer(T83);
ClearTimer(T84);
ClearTimer(T85);
ClearTimer(T86);
ClearTimer(T87);
ClearTimer(T88);
ClearTimer(T89);
ClearTimer(T90);
ClearTimer(T91);
ClearTimer(T92);
ClearTimer(T93);
ClearTimer(T94);
ClearTimer(T95);
ClearTimer(T96);
ClearTimer(T97);
ClearTimer(T98);
ClearTimer(T99);
ClearTimer(T100);
    
```

The VEX has 4 built in timers T1, T2, T3, and T4. But if you need to reset and start Timer T1, you would type:

```

ClearTimer(T1);
    
```

Then, you can retrieve the value of the timer by using time1(T1), time2(T2), or time3(T3) 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). While the value of the timer is less than 2 seconds, the robot will move forward until touch and then turn. The program ends after 2 seconds.

© Copyright RobotC Systems | For use with ROBOTC Systems

Reference

Potentiometers Overview

The Potentiometer is used to measure the angular position of the axle or shaft passed through its center. The center of the sensor can rotate roughly 345 degrees and outputs values ranging from 0-1023 to the VEX PC and 0-4095 to the VEX Cortex.

The Potentiometer can be attached to the robot using the mounting pins surrounding the center of the sensor. The only possible flexibility for the connection of the Potentiometer, utilizing the full range of motion to be obtained more easily.

When mounted on the rotating shaft of a moving portion of the robot, such as an arm or gripper, the Potentiometer provides precise feedback regarding its angular position. This sensor data can then be used for accurate control of the robot.

Mounted Potentiometer
The potentiometer is mounted on the robot using the mounting pins surrounding the center of the sensor.

CAUTION! When mounting the Potentiometer on your robot, ensure that the range of motion of the rotating shaft does not exceed that of the sensor. Failure to do so may result in damage to your robot and the Potentiometer.

Clear it Up
The range of motion of the rotating shaft should be checked to ensure the sensor is not damaged.

Note: For sensor feedback, use the sensor feedback.

© Copyright RobotC Systems | For use with ROBOTC Systems

Reference

LED Overview and Natural Language Sample Code

The VEX LEDs are treated as digital output devices, and use digital pins on the Cortex. By having different LEDs on a off during program execution, they can provide useful feedback for commands obtaining feedback from your robot while it is running.

LED ON After Touch
This code has the robot turn on an LED, and then, when the touch sensor is pressed, turn the LED off.

```

task main ()
{
  startMotor(motor1); //Turn on LED on Digital Port 13
  wait1Touch(digital12); //Wait until the Touch sensor is Digital // Port 12 is pressed
  startMotor(digital11); //Turn off LED on Digital Port 13
}
    
```

© Copyright RobotC Systems | For use with ROBOTC Systems



Natural Language Introductory Video -The Video at the left and all of the handouts are posted for free at: www.robotc.net/NaturalLanguage/

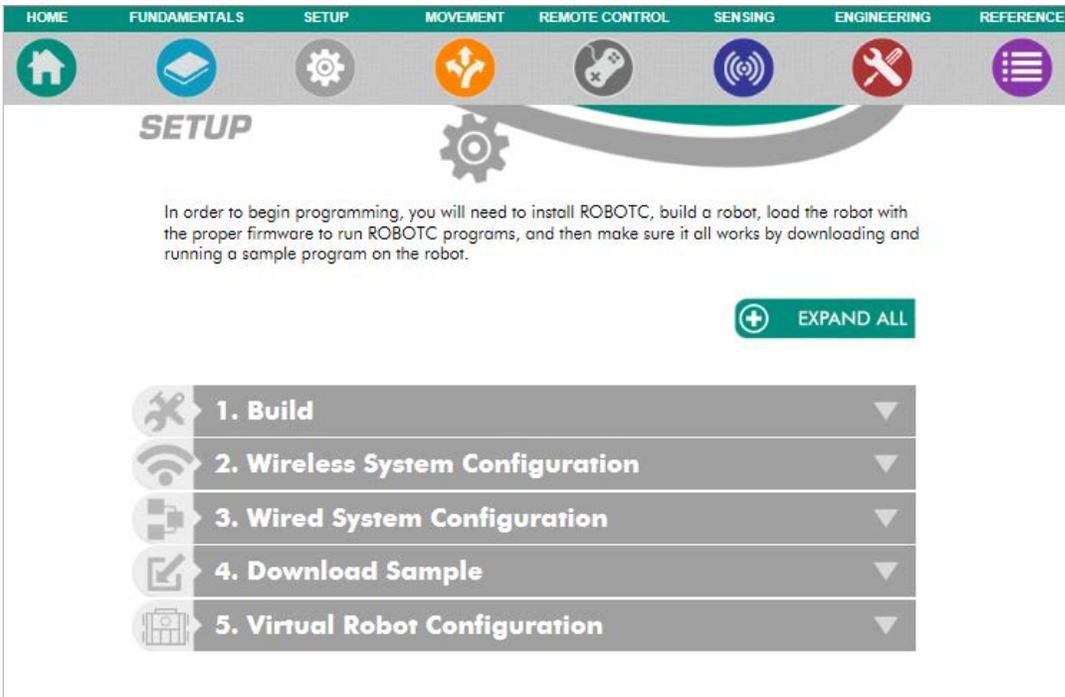
VEX Cortex Setup

The screenshot shows the ROBOTC software interface with the 'SETUP' section selected. The navigation bar at the top includes: HOME, FUNDAMENTALS, SETUP (highlighted), MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. Below the navigation bar, the 'SETUP' section is titled with a gear icon. A paragraph of text reads: 'In order to begin programming, you will need to install ROBOTC, build a robot, load the robot with the proper firmware to run ROBOTC programs, and then make sure it all works by downloading and running a sample program on the robot.' Below this text is a green button with a plus icon and the text 'EXPAND ALL'. A list of five steps is displayed, each with an icon and a dropdown arrow:

1. Build
2. Wireless System Configuration
3. Wired System Configuration
4. Download Sample
5. Virtual Robot Configuration

Setup - In order to begin programming, you will need to build a robot, use ROBOTC to update the firmware on the VEX Cortex and VEXnet Joystick, and then make sure that it works by downloading and running a sample program. The SETUP section guides a user step-by-step through this process. This section is divided into three sections: Build, System Configuration, and Download Sample.

Setup/Build



HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING ENGINEERING REFERENCE

SETUP

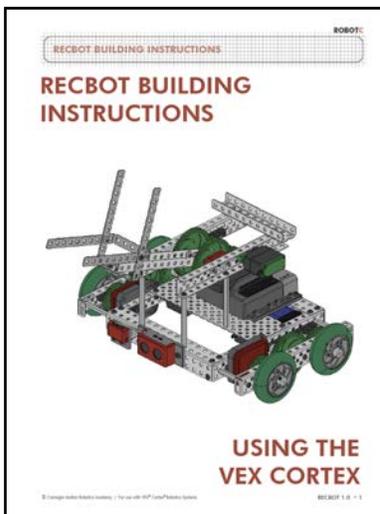
In order to begin programming, you will need to install ROBOTC, build a robot, load the robot with the proper firmware to run ROBOTC programs, and then make sure it all works by downloading and running a sample program on the robot.

EXPAND ALL

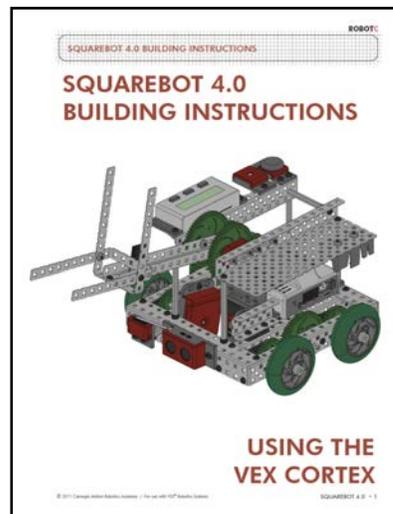
1. Build
2. Wireless System Configuration
3. Wired System Configuration
4. Download Sample
5. Virtual Robot Configuration

Note: the Clawbot with Sensors build on the next page is the recommended build for schools that purchased Clawbot kits.

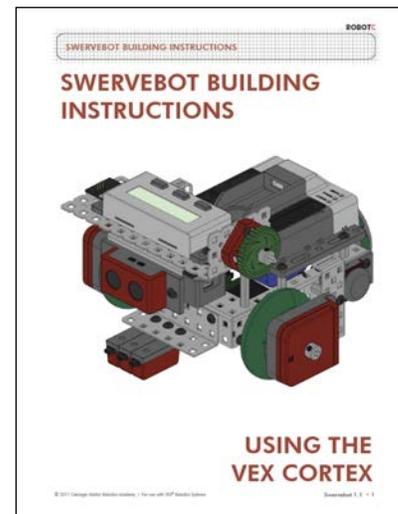
Robot Building instructions come in PDF format allowing you to either print them or use them directly from the screen. Additional instructions and updates will always live on the VEX Building Link at the Robotics Academy website.



RECBOT is the robot used most in the curriculum. It can be built out of the VEX Protobot kits with plus sensor kits.

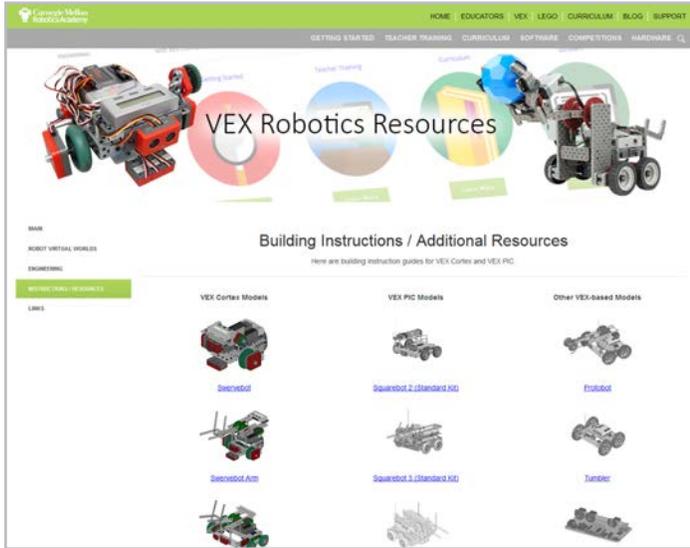


Squarebot 4.0 is an excellent alternative to the RECBOT, and takes up a little less space. It requires very little adjustment to make it work with the videos, and is also buildable using Protobot kits plus sensor kits.



Swervebot is a significantly smaller robot model that requires significant cutting to the VEX metal to create. Its smaller form factor make it ideal for storage and for robot applications that require sharp turns such as line following.

Additional Building Instructions for VEX Robots

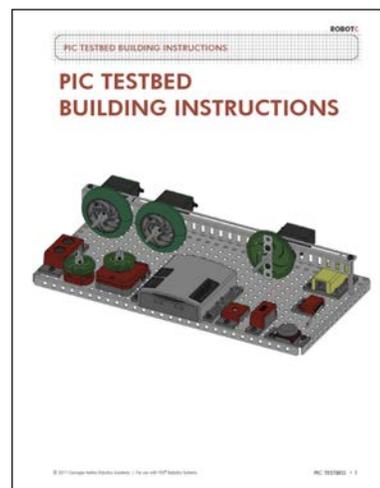
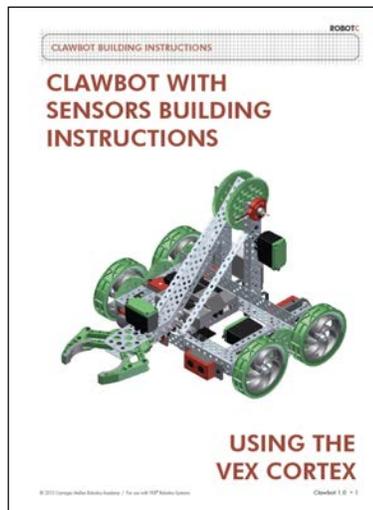
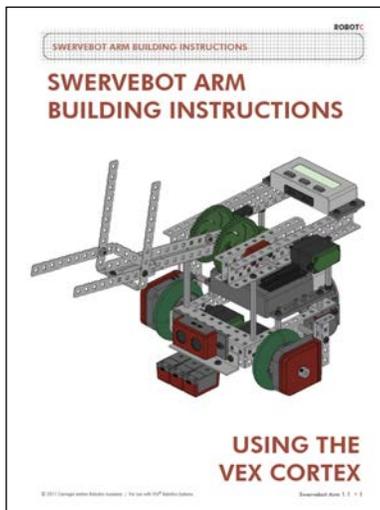


Choosing a robot model is a big choice because you may not have the time to have kids build robots multiple times during the year.

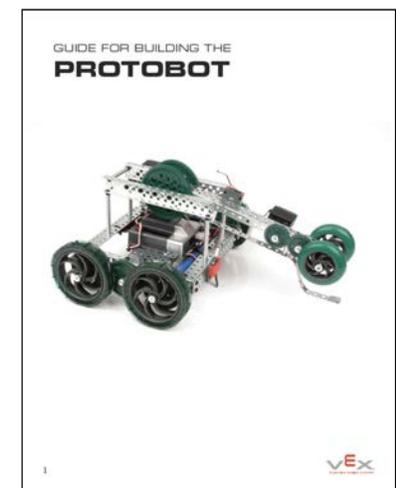
Read all of the notes below each set of plans before you begin.

All of the Robot Plans plans are available via the VEX Robotics link at the Robotics Academy website.

www.vexteacher.com



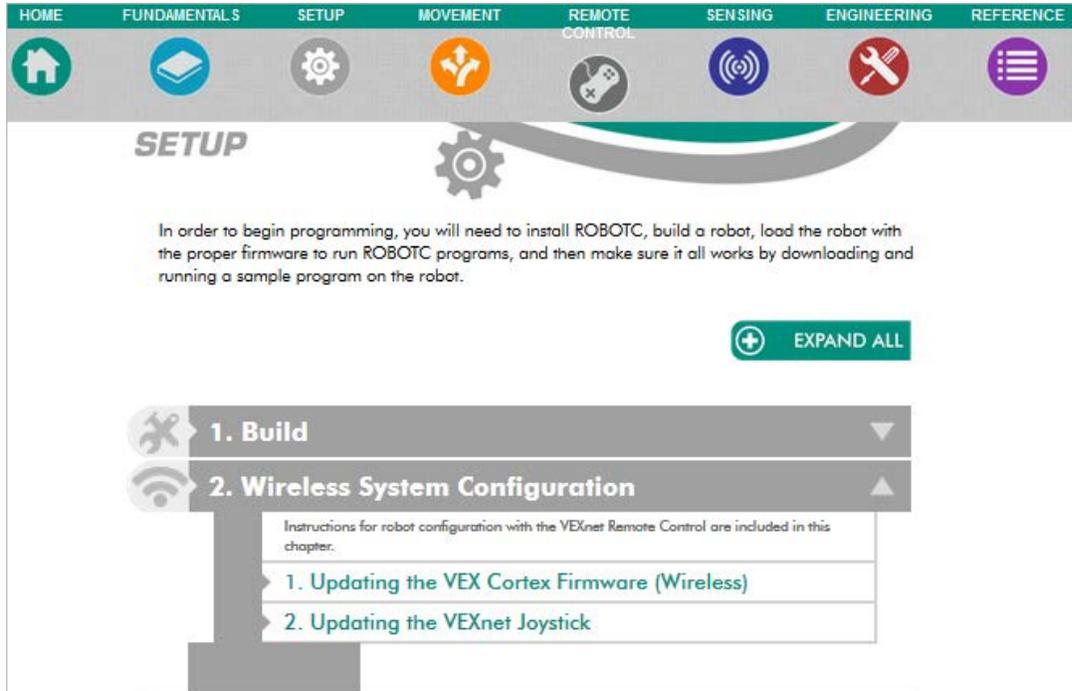
The Clawbot instructions are the best choice if you have the VEX Clawbot kits. Some adaptation will be needed on your part to account for the differences between it and the other builds, such as which motors are used to move the robot forward.



The Test Bed works well with Natural Language programming, and allows the system to be learned in a controlled environment.

Tumbler and Protobot are excellent robots but are not recommended for use with the curriculum.

Setup/Wireless System Configuration

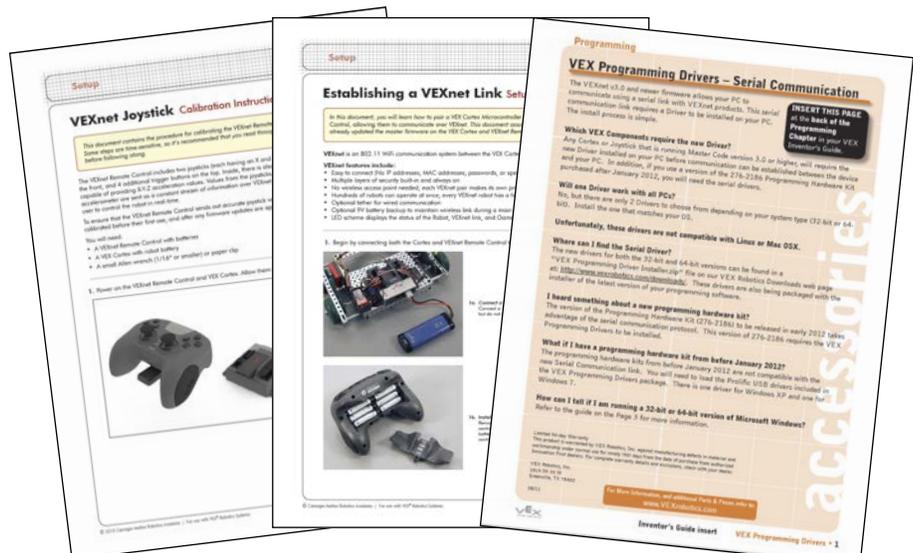


Instructions for Wireless Programming

Updating the Cortex Firmware Video - Teaches students about the VEX Master firmware and the ROBOTC user firmware. The video teaches students how to download the latest firmware and software before they begin programming their robots.

Updating the VEX Joystick Firmware Video - Guides students step-by-step as they update the VEXnet Joystick firmware.

Establishing a VEXnet Link, VEXnet Joystick Calibration, and VEX Programming Drivers Reference Guides - Step-by-step instructions that guide students through the VEXnet setup and configuration process.



Setup/Wired System Configuration

HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING ENGINEERING REFERENCE

SETUP

In order to begin programming, you will need to install ROBOTC, build a robot, load the robot with the proper firmware to run ROBOTC programs, and then make sure it all works by downloading and running a sample program on the robot.

COLLAPSE ALL

- 1. Build
- 2. Wireless System Configuration
- 3. Wired System Configuration
 - Instructions for robot configuration with USB Cable are included in this chapter.
 - 1. Updating the VEX Cortex Firmware (Wired)
- 4. Download Sample
- 5. Virtual Robot Configuration

FUNDAMENTALS PREV. NEXT MOVEMENT

VEX Cortex Video Trainer using ROBOTC
Copyright(c)2014 Carnegie Mellon Robotics Academy. All rights reserved.

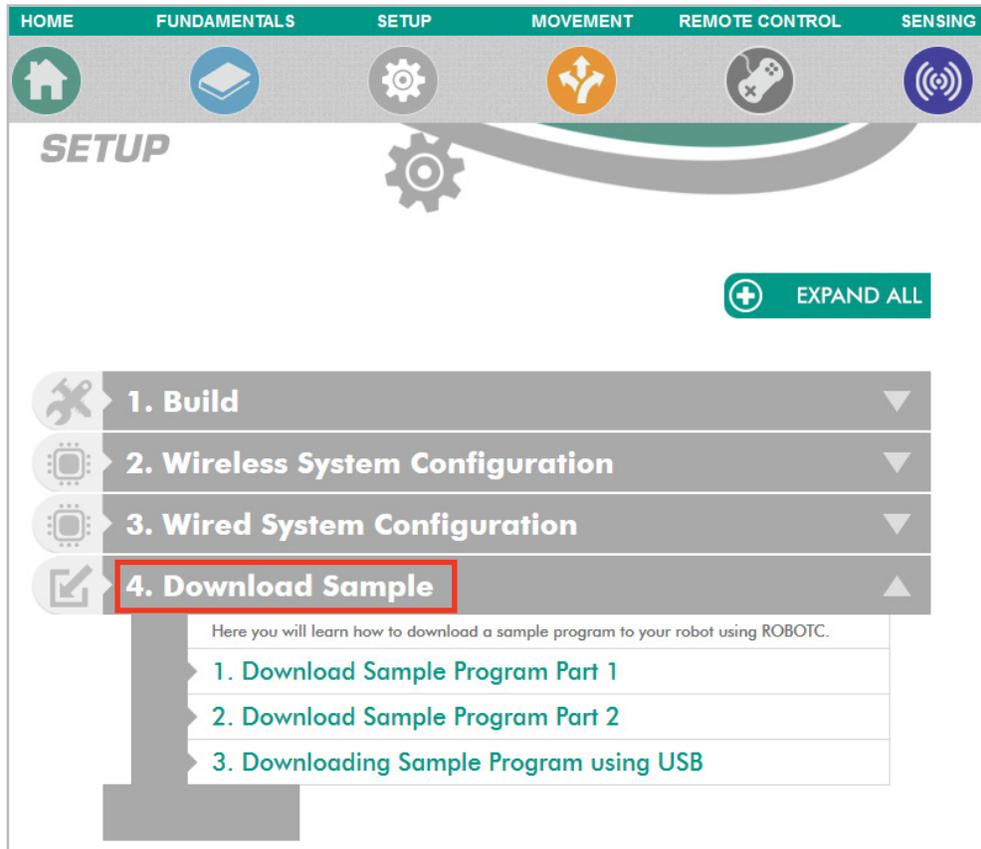
Instructions for Wired System Configuration

Updating the Cortex Firmware Video - Teaches students about the VEX Master firmware and the ROBOTC user firmware. The video teaches students how to download the latest firmware and software before they begin programming their robots.

VEX Programming Drivers Reference Guides - Step-by-step instructions that guide students through the VEXnet setup and configuration process.



Setup/Downloading Sample Program



Instructions for Wireless Programming

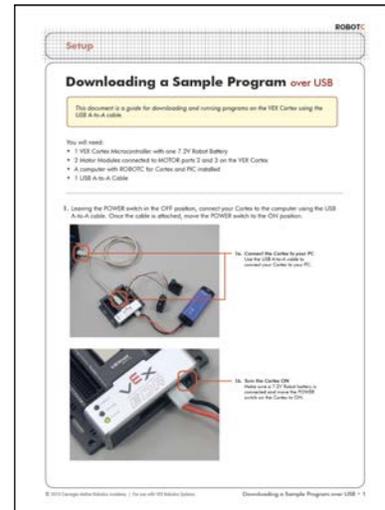
Downloading a Sample Program over VEXnet Videos - Guides students step-by-step through the process of downloading a ROBOTC program. The first video tells them how to setup their hardware. The second video shows them how to powerup their VEX Cortex and VEXnet Joystick, confirm their settings, and then download their program. Once this process is complete the Cortex remembers all of the settings and so setup only need to be done once.

USB to Serial Driver Installation and Downloading a Sample Program over USB Reference Guides - Step-by-step instructions that guide students through installing drivers.

Download Sample Program over VEXnet Part 1



Download Sample Program over VEXnet Part 2



ROBOTC Has over 250 Working Sample Programs

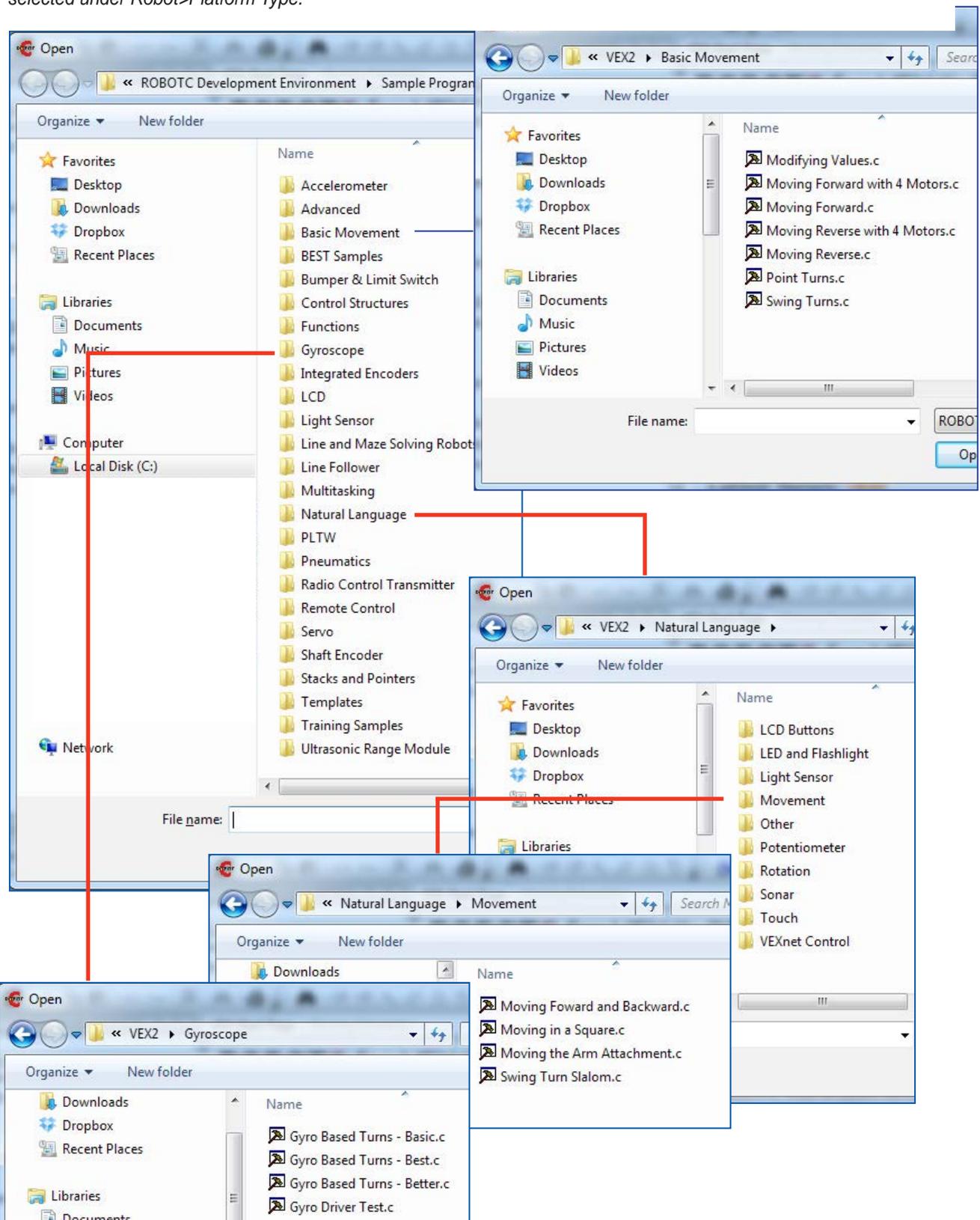
Often it is easier for new programmers to get a feel for programming by editing working programs. To access ROBOTC's extensive library of examples select "File" and then "Open Sample Program". Each folder contains examples of working code that can be loaded onto a student's robot and tested. *Note: the code in those sample programs was written for a specific robot and adjustments may need to be made in ROBOTC's Motors and Sensors Setup.*

The screenshot shows the ROBOTC software interface. The 'File' menu is open, and the 'Open Sample Program' option is highlighted with a red box. A file explorer window is also open, showing a directory structure of sample programs. The file explorer window is titled 'Open' and shows the path 'ROBOTC Development Environment > Sample Programs > VEX2'. The file explorer window lists various folders such as 'Accelerometer', 'Advanced', 'Basic Movement', 'BEST Samples', 'Bumper & Limit Switch', 'Control Structures', 'Functions', 'Gyroscope', 'Integrated Encoders', 'LCD', 'Light Sensor', 'Line and Maze Solving Robots', 'Line Follower', 'Multitasking', 'Natural Language', 'PLTW', 'Pneumatics', 'Radio Control Transmitter', 'Remote Control', 'Servo', 'Shaft Encoder', 'Stacks and Pointers', 'Templates', 'Training Samples', and 'Ultrasonic Range Module'.

1. Select "File"
2. Select "Open Sample Program"
3. Navigate to the folder that you want to open

ROBOTC Has over 250 Working Sample Programs

Pictured below are the types of programs that a student will find. The programs range from Basic Movement using standard ROBOTC to Basic Movement using ROBOTC's Natural Language Library to using the VEX Gyro Sensor and everything in between. *Note: when programming using Natural Language Commands, Natural Language must be selected under Robot>Platform Type.*



Setup/Virtual Robot Configuration

The screenshot shows the ROBOTC software interface with a navigation bar at the top containing icons for HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. The 'SETUP' menu is expanded, showing a list of options: 1. Build, 2. Wireless System Configuration, 3. Wired System Configuration, 4. Download Sample, and 5. Virtual Robot Configuration. The 'Virtual Robot Configuration' option is selected, revealing a sub-menu with three items: 1. Downloading your First Program (Virtual), 2. Camera Operation in Robot Virtual Worlds, and 3. The Measurement Toolkit. A 'COLLAPSE ALL' button is visible above the menu. At the bottom of the interface, there are navigation buttons for 'FUNDAMENTALS', 'PREV.', 'NEXT', and 'MOVEMENT', along with a copyright notice: 'VEX Cortex Video Trainer using ROBOTC Copyright(c)2014 Carnegie Mellon Robotics Academy. All rights reserved.'

This lesson set includes three instructional videos designed to help the new user to program using Robot Virtual Worlds.

The thumbnail features a green and grey header with a gear icon and the ROBOTC logo. The main text reads 'SETUP Downloading your First Program (Virtual)'. Below this, it states 'IN THIS LESSON YOU WILL LEARN' followed by a bullet point: '• How to load a program to a virtual robot.' The bottom of the thumbnail includes a copyright notice: '© Copyright 2013 Carnegie Mellon Robotics Academy' and a video player control bar.

This block contains two video thumbnails. The top one is titled 'Camera Operation in RWV' and has a sub-header 'Camera Controls in the Virtual Worlds'. The bottom one is titled 'Measurement Toolkit' and features a background of technical drawings and a ruler. It includes the text 'VIRTUAL MEASUREMENT TOOLS' and 'Great new measurement tools included with RWV.'

Robot Virtual World Software - the Curriculum Companion



Teaching Programming via Robot Virtual Worlds (RVW)

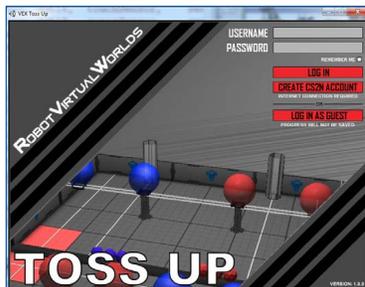
The Curriculum Companion is designed to mirror the VEX Cortex Video Trainer using ROBOTC. The RVW software includes over 40 programming challenges that are found in the VEX Cortex Video Trainer Curriculum.

Throughout the Introduction to Programming section you will find icons that show which challenges are included.

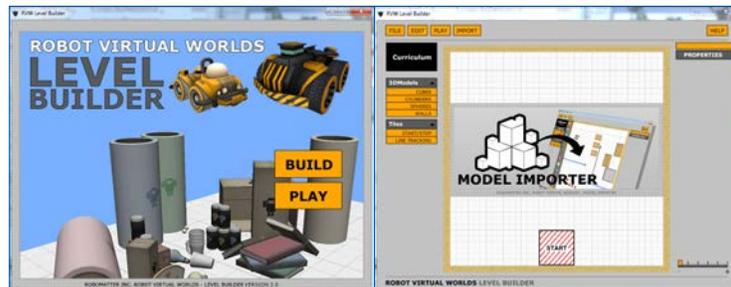
Additionally, students can pick from five robot configurations to choose to program, they are shown at the left of this page; VEX Squarebot, VEX Clawbot, VEX Swervebot, BuggyBot, and Mammal Bot.

Additional VEX Robotics Virtual Worlds

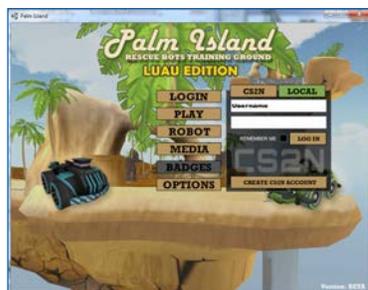
A ROBOTC RVW License gives you access to the Curriculum Companion, *PLUS* all of the RVWs below.



The current VEX Robotics Engineering Challenge. Students can begin developing code and strategies before they build their robots



The Level Builder and Model Importer allow student to build their own levels and import their own models into their new worlds. RVW is compatible with any modeling software that can generate a .STL file, including Autodesk, SolidWorks, and Google Sketchup.



The Palm Island, Operation Reset, and Atlantis Robot-To-The-Rescue programming games take kids to yet to be discovered worlds where their robots need to be programmed to solve challenges!

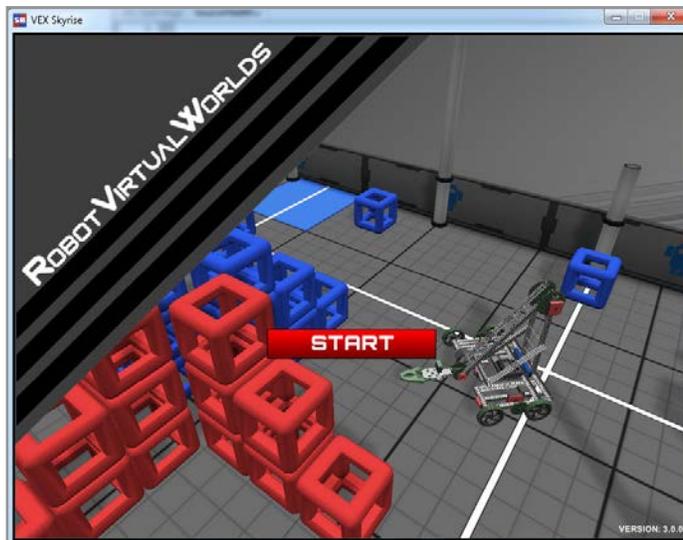
Virtual Robot Competitions

Each year, VEX robot competitions are modeled and available for use in your class. Pictured below are prior year example competitions.



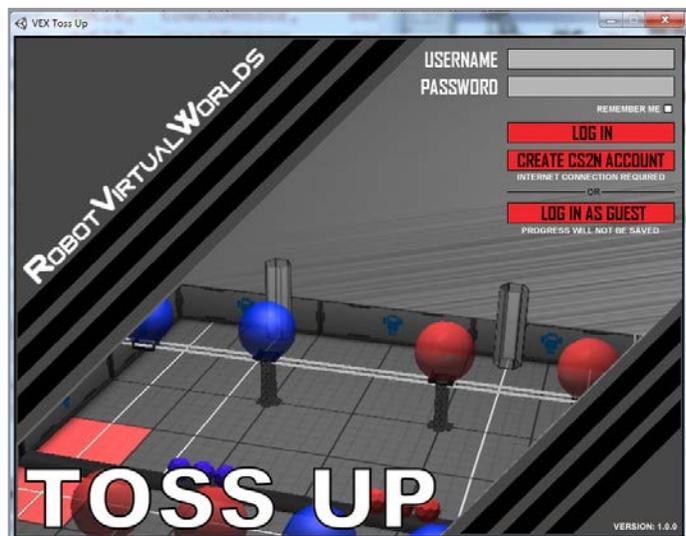
Host your own in school virtual competition! All of these games are available at the Robot Virtual World website.

Pictured above is Nothing But Net, the VEX 2015 Game



Pictured at the right is Skyrise, the VEX 2014 Game

Pictured at the right is Toss UP, the VEX 2013 Game



Guided Programming and Engineering Challenges

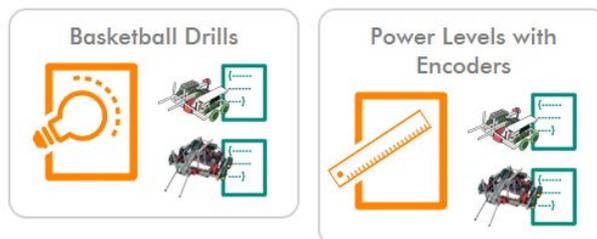
The screenshot displays the VEX Cortex Video Trainer interface, which is organized into a grid of lesson sets. At the top, a navigation bar includes icons for HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. Below this, four lesson sets are visible, each with a title, a brief description, and a list of challenges. The 'MOVEMENT' lesson set includes challenges like '1. Challenge', '2. Moving Forward', '3. Speed and Direction', '4. Shaft Encoder', '5. Automated Straight', and '6. Integrated Encoders'. The 'REMOTE CONTROL' lesson set includes '1. Challenge', '2. Joystick Mapping', '3. Timers', and '4. Buttons'. The 'SENSING' lesson set includes '1. Challenge', '2. Limiting the Arm', '3. Behaviors and Functions', '4. Forward until Near', '5. Line Tracking', '6. Turn for Angle', and '7. Using the LCD'. The 'ENGINEERING' lesson set includes '1. Safety', '2. VEX Hardware', '3. Engineering Process', '4. Competition Programming', and '5. Rubrics'. Each lesson set also features an 'EXPAND ALL' button and a 'VIEW' button.

The Guided Programming and Engineering Lessons are designed to be “learner centered”.

Students work through the lessons in a step-by-step fashion. The first three units: Movement, Remote Control, and Sensing contain Lesson Sets that teach a particular programming concept and include several programming and engineering challenges (see below). The last lesson set is “Engineering”. This section includes resources that students will use as they solve their engineering design challenges.

Engineering Investigations and Programming Challenges

At the bottom of many of the lessons students will find programming challenges and engineering investigations. The Basketball Drills picture shows the icon that indicates a programming challenge, the Power Levels with Encoders icon represents an Engineering Investigation.



The Movement Unit

The screenshot displays the ROBOTC software interface. At the top, a navigation bar includes tabs for HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. Below this bar is a row of icons corresponding to each tab. The 'MOVEMENT' tab is highlighted with an orange arrow pointing to it. Below the navigation bar, the word 'MOVEMENT' is written in large, bold, orange letters. To the right of this text is a green button with a white plus sign and the text 'EXPAND ALL'. Below the 'EXPAND ALL' button is a list of six items, each with an icon on the left and a dropdown arrow on the right:

- 1. Challenge
- 2. Moving Forward
- 3. Speed and Direction
- 4. Shaft Encoder
- 5. Automated Straightening
- 6. Integrated Encoders

The Movement Unit

The Movement Unit is taught using four 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 an “engineering lab” 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, Shaft Encoders, begins to introduce students to Boolean Logic and While Loops. Automated Straightening introduces students to if-else Statements and Variables and teaches them to develop their own automated straightening algorithm. The Integrated Encoders lesson teaches how to use the Integrated Motor Encoders.

The Movement Unit also includes fourteen programming challenges where students are challenged to solve simple movement programming challenges. Although some of the challenges appear to be repetitive, the extra challenges enable the teacher to differentiate instruction, it will be up to the teacher to decide which students do which challenges. The extra challenges give the teacher to differentiate the instruction based on student’s ability.

Note: It will be important to remind students that although the initial work may seem easy, that the skills that they learn in the movement unit are foundational pieces that they must understand before they move to the Remote Control and Sensing Units.

Movement/Challenge Description

Each of the large units, Movement; Remote Control, and Sensing, has a unit programming challenge that students will be guided step-by-step toward a solution.

1. Challenge

Your first project-based challenge will be a maze called "The Labyrinth". Successful completion of this challenge will require a thorough understanding of the movement behaviors and commands available in ROBOTC.

Beware! Not all versions of the Labyrinth are the same. Your board, your robot, and your program may all differ from the ones shown in the video... now is your time to shine as a problem-solver and programmer!

1. Labyrinth Challenge

2. Moving Forward

Labyrinth Programming Challenge

Each Programming Unit (Movement, Remote Control, Sensing, and Resources) contains a Unit 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".

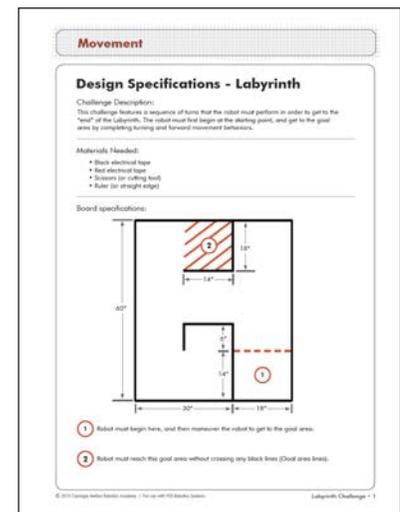
In this challenge, students will learn:

- 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

The Movement Unit programming challenge comes with a PDF that explains the rules to the challenge, an example video solution.

The Labyrinth Challenge is also available in a Robot Virtual World simulation format.



Movement/Moving Forward



The Moving Forward Lesson Set

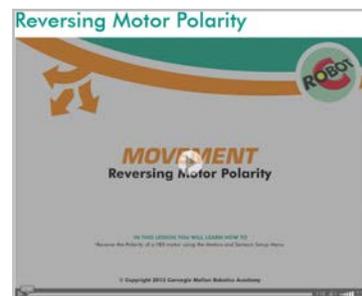
Each Lesson Set is designed to teach a related set of programming concepts and is designed to give students confidence opening a program and understanding what the individual lines of code mean. Every set is supported with a combination of video and print resources:

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

Reversing Polarity Video - This video shows students how to make their robot change directions and turn.

Renaming Motors Video - ROBOTC provides a very handy utility that allows the programmer to rename their motors using names that make sense to them (i.e. leftMotorArm, or RightWheel). This video shows how it works.

The Moving Forward Lesson Set is continued on the next page.



Movement/Moving Forward (continued)



Programming Challenge

Wait States

Investigation Summary:

Part 1
Download and run the sample code below on your robot. Observe how far the robot travels. Then modify the amount of time that the robot travels by changing the value of line 3 from 500 to 1000. Download and run the program and observe the distance the robot travels. Repeat these steps for values of 2500 and 3200 as well. In each case, the variable that you are manipulating is the amount of time that the robot "waits" until it does something else.

```

Sample Code
1 // Wait 500 ms
2 motor[leftMotor] = 127;
3 wait(500);
4 motor[rightMotor] = 127;
5 wait(500);
6

```

Part 2
Measure the average distance that the robot travels in 500 milliseconds, using three attempts:
Attempt 1:
Attempt 2:
Attempt 3:
Average Distance:
Use the Average Distance you found to predict how far the robot will travel for 1000, 2500, and 3200 milliseconds. Fill in the chart with your predictions.

Wait state in milliseconds	Wait state in seconds	Predicted distance traveled
500	1/2	
1000		
2500		
3200		

Reference

Running a Program

Once a program has been successfully written, it needs to be given to the robot to run. The following steps will guide you through the process of downloading your program to the robot, and then running it remotely or connected to your computer.

1. Verify that your VEX Cortex is turned off. Make sure it's not powered by the battery or PC through the USB cable.
2. If you are programming over USB, connect the Cortex to the PC using the USB A-to-A cable. This will partially power the Cortex, then turn the Cortex on to fully power it using the battery.
Note: Your Robot - VEX Cortex Download Method should be set to "Download Using USB Only" if you are not using the wireless V5Direct connection.
3. Click "Robot" on the top menu bar of the ROBOTC window, and select "Compile and Download Program".
4. You may be prompted to save your program. If so, save it in the same directory as your other programs.
5. If there are errors in your code the compiler will identify them for you and you will need to correct them before a successful download can be completed.

Programming Challenge

Sumo Bot

Relational Targon vs. Power Level Investigation Description
In this investigation you will determine if there is a proportional relationship between the robot's motor power level and its ability to push items out of a ring. Program your robot to move at 1%, 1%, 1%, 1% to enter full power levels and record the maximum number of cans it is able to move out of the ring.

Materials Needed

- Black electrical tape
- Scissors (or cutting tool)
- 20 Soda cans (or something similar)

Robot Specifications

Note: Diagrams are not drawn to scale.

Chart

Power Level	Number of Cans
127	
86	
43	
21	

To show a proportional relationship between power level and the number of cans that the robot can push out of the ring! Please describe your findings.

The Moving Forward Lesson Set

Moving Forward Timing Video - Timing is the least accurate way to control a robot, but it is also the simplest method.

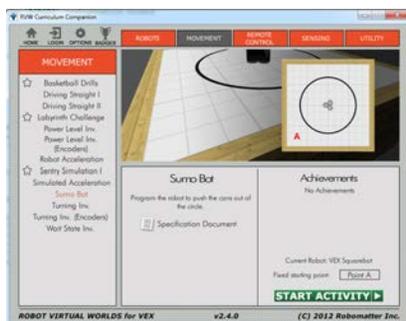
Wait States Engineering Investigation - This PDF contains an Engineering Investigation that requires students to take an initial measurement and then to make predictions of how far their robot will travel based on changing the amount of time the robot is set to run.

SumoBot Challenge - At this point in a student's programming career all they can do is move forward. This is a nice engineering challenge that allows students to focus on weight distribution, traction, and design.

Robot Virtual World Simulations

Wait States Engineering Investigation - Students are required to run their robots, take measurements, and then predict and record their measurements for their next run.

This SumoBot Challenge - Enables students to simulate the programming challenge.



Movement/Speed and Direction

HOME
FUNDAMENTALS
SETUP
MOVEMENT
REMOTE CONTROL
SENSING
ENGINEERING
REFERENCE

MOVEMENT

+ EXPAND ALL

1. Challenge
▼

2. Moving Forward
▼

3. Speed and Direction
▲

1. Motor Power Levels
▶

2. Turn and Reverse
▶

3. Manual Straightening
▶

Modifying motor commands to move in different ways.

1. Motor Power Levels
▶

2. Turn and Reverse
▶

3. Manual Straightening
▶

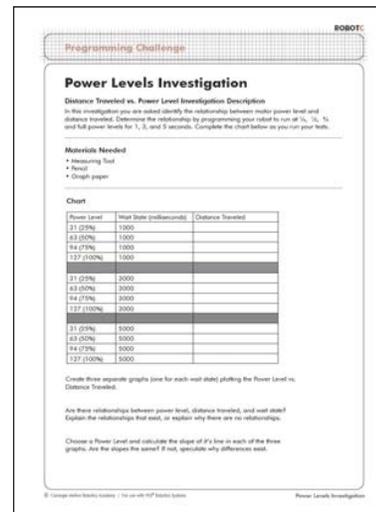
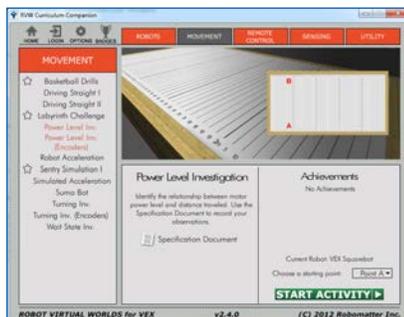
The Speed and Direction Lesson Set

In this lesson set students continue to build confidence in their ability to program simple movements. The lesson set consists of three videos, three Engineering Challenges, and two Programming Challenges.

Motor Power Lesson

The Motor Power Level Video - 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 "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 will investigate if there is a proportional relationship between power levels and speed.

The Power Level Investigation is also available as a Robot Virtual World Simulation.



Movement/Speed and Direction (continued)

Speed and Direction

Turn and Reverse Video - teaches students how changing motor polarity enables the robot to turn and change direction.

Manual Straightening Video - Challenges the student to use power levels to control the robot to go straight (students will learn a better method soon).

Programming Challenge PDFs - The Power Level Investigation, Driving Straight Challenge, and Sentry Simulation provide students with opportunities to practice programming and learn about the VEX robot system.



Engineering Lab

Sentry Simulation Level One

Challenge Description
Your company is commissioned to design a robot to patrol around the outside of a building. You are the lead engineer and tasked to develop a "brand of concept" prototype that can automatically roam around an object.

Questions:

- You are required to demonstrate a solution that uses swing turns.
- You are required to demonstrate a solution that uses point turns.
- The robot will turn 1° absolute PVC gates at the corners of the object and the 4° half-degree if on a turn or piece of PVC or leave the robot's robot at the object. (see the diagram below)

Materials Needed

- Eight pieces of 1" diameter PVC
- One robot

Robot Specifications

Note: Diagrams are not drawn to scale.

Were the robot's turns more reliable when using point turns or swing turns? Why?

What would happen to your program if you changed the wheel diameter of your robot?

What happens as the robot's battery power changes?

Programming Challenge

Power Levels Investigation

Distance Traveled vs. Power Level Investigation Description
In this investigation you are asked to identify the relationship between motor power level and distance traveled. Determine the relationship by programming your robot to run at 1%, 10%, 20%, and full power levels for 1, 2, and 3 seconds. Complete the chart below as you run your tests.

Materials Needed

- Measuring Tool
- Robot
- Graph paper

Chart

Power Level	Wheel Size (diameter)	Distance Traveled
21 (25%)	1000	
43 (50%)	1000	
64 (75%)	1000	
127 (100%)	1000	
21 (25%)	2000	
43 (50%)	2000	
64 (75%)	2000	
127 (100%)	2000	
21 (25%)	5000	
43 (50%)	5000	
64 (75%)	5000	
127 (100%)	5000	

Create three separate graphs (one for each wheel size) plotting the Power Level vs. Distance Traveled.

Are there relationships between power level, distance traveled, and wheel size? Explain the relationships that exist, or explain why there are no relationships.

Choose a Power Level and calculate the slope of the line in each of the three graphs. Are the slopes the same? If not, speculate why differences exist.

Engineering Lab

Driving Straight Challenge

Problem
It is not easy to get your robot to drive straight without feedback from sensors. There are several variables that come into play: your robot may not be equally weighted, there may be more friction on one side than the other, the robot's weight distribution may be uneven, there may be unbalanced axles. Eventually you will have feedback from encoders to help align the movement of your robot, but until then you will have to program your robot manually. Adjust the power levels of each motor independently to get your robot to drive straight. If the robot drifts to the right, that means that your right motor is driving more slowly than your left motor, so you should speed your right motor up.

Challenge Description
Using wide cones or small objects as "roadway cones", create a long driveway for your robot to more than 3" wider than the actual robot. Program the robot so that it can drive from one end to the other without bumping a single cone or object. The robot that can travel the furthest down the driveway without hitting a cone is the winner.

Challenge Specifications

Robot Virtual World Simulations

Motor Power Levels, Sentry Simulation, Turning, and the Driving Straight challenges are available via the ROBOTC RVW simulation software.

Engineering Lab

Turning Investigation

Investigation Description
In this investigation we will complete calculations for swing turns and point turns.

Swing turns - In this type of turn the set of wheels or tracks on one side of the robot are powered and the other set of wheels or tracks remain off.

Point turns - In this type of turn the set of wheels or tracks on one side of the robot are powered forward and the other set is powered in reverse, causing the robot to turn on its central axis.

Procedures:

Write a program that allows your robot to execute a 90 degree swing turn. At this point in your programming screen, all that you know how to adjust a timing and power levels. Research, test and improve your program until you produce an exact 90 degree turn.

Swing Turn Pseudocode

- Power the right motor in the forward direction for an amount of time.
- Also apply no power (or brake) the left wheel for the amount of time.
- Turn both motors off.

Now write a program that allows your robot to complete a 90 degree point turn. Research, test and improve your program until your robot turns exactly 90 degrees.

Now that you have the basics, complete the investigation by completing the table on the next page. Test each distance multiple times to ensure that your results are repeatable.

Point Turn Pseudocode

- Power the right motor in the forward direction for an amount of time.
- Also power the left motor in the reverse direction for the amount of time.
- Turn both motors off.

Answer the following questions. Use the back of this paper or another sheet of paper if you need more space to write your answers.

What variables did you adjust for the robot to execute a 90 degree swing turn? If you mathematically manipulated the time variables, did you get the results that you expected? If you disabled the amount of time, did your robot turn 90 degrees? Or did the robot turn more or less?

What variables did you adjust to make your 90 degree point turn? If you mathematically manipulated those variables, did you get the results that you expected?

Did you try adjusting the variable power during the investigation? What happened when you increased the power level to a higher number? Did this increase or decrease your robot's turning accuracy?

Which type of turn type is more reliable point turns or swing turns? Why?

Which type of turn is faster?

When you complete the table below on the next page, graph your results. Is there a proportional relationship between time and angle turned or power and time? Be prepared to present your results.

ROBOT VIRTUAL WORLDS for VEX v2.4.0 (C) 2012 Robomatter Inc.

MOVEMENT

- Robotball Drills
- Driving Straight I
- Driving Straight II
- Labyrinth Challenge
- Power Level Inv.
- Power Level Inv. (Encoder)
- Robot Acceleration
- Sentry Simulation I
- Simulated Acceleration
- Sumo Bot
- Turning Inv.
- Turning Inv. (Encoder)
- Wall State Inv.

Turning Investigation

Write programs that cause the robot to perform different point turns and swing turns. Use the Specification Document to record your observations.

Achievements

No Achievements

Current Robot VEX Squabbot

Feed starting point

START ACTIVITY

ROBOT VIRTUAL WORLDS for VEX v2.4.0 (C) 2012 Robomatter Inc.

MOVEMENT

- Robotball Drills
- Driving Straight I
- Driving Straight II
- Labyrinth Challenge
- Power Level Inv.
- Power Level Inv. (Encoder)
- Robot Acceleration
- Sentry Simulation I
- Simulated Acceleration
- Sumo Bot
- Turning Inv.
- Turning Inv. (Encoder)
- Wall State Inv.

Sentry Simulation I

Program the robot to patrol the perimeter of an object. Touching the object will result in failure. Completion of the challenge is required for the Movement Mastery badge.

Achievements

Sentry Starter

Sentry Simulation Completion

Current Robot VEX Squabbot

Feed starting point

START ACTIVITY

ROBOT VIRTUAL WORLDS for VEX v2.4.0 (C) 2012 Robomatter Inc.

MOVEMENT

- Robotball Drills
- Driving Straight I
- Driving Straight II
- Labyrinth Challenge
- Power Level Inv.
- Power Level Inv. (Encoder)
- Robot Acceleration
- Sentry Simulation I
- Simulated Acceleration
- Sumo Bot
- Turning Inv.
- Turning Inv. (Encoder)
- Wall State Inv.

Driving Straight I

From the starting line, program the robot so that it travels from one end to the other without bumping into single object.

Achievements

No Achievements

Current Robot VEX Squabbot

Feed starting point

START ACTIVITY

ROBOT VIRTUAL WORLDS for VEX v2.4.0 (C) 2012 Robomatter Inc.

MOVEMENT

- Robotball Drills
- Driving Straight I
- Driving Straight II
- Labyrinth Challenge
- Power Level Inv.
- Power Level Inv. (Encoder)
- Robot Acceleration
- Sentry Simulation I
- Simulated Acceleration
- Sumo Bot
- Turning Inv.
- Turning Inv. (Encoder)
- Wall State Inv.

Power Level Investigation

Identify the relationship between motor power level and distance traveled. Use the Specification Document to record your observations.

Achievements

No Achievements

Current Robot VEX Squabbot

Feed starting point

START ACTIVITY

Movement/Shaft Encoders

The screenshot shows the ROBOTC curriculum interface. At the top, there is a navigation bar with tabs for HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. Below the navigation bar, the 'MOVEMENT' section is highlighted in orange. A large orange arrow points from the 'MOVEMENT' tab to the '4. Shaft Encoder' lesson. The '4. Shaft Encoder' lesson is expanded, showing a list of sub-lessons: 1. Shaft Encoders, 2. Forward Until Distance Part 1, 3. Forward Until Distance Part 2, 4. The Sensor Debug Window, and 5. Forward and Turning. The first sub-lesson, '1. Shaft Encoders', is highlighted in orange.

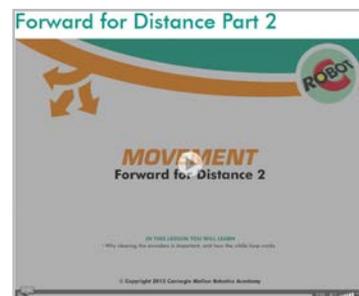
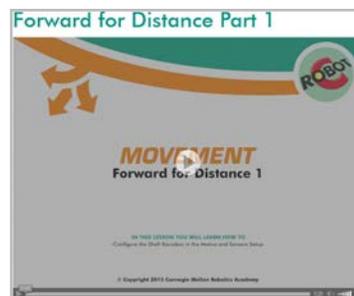
Shaft Encoders Lesson Set

Shaft encoders allow you to accurately control the distance your robot travels. Since the release of this curriculum VEX Robotics released smart motors where the encoder is built into the motor. You can find information about the Integrated Motor Encoders: http://www.robotc.net/wiki/Tutorials/Programming_with_the_new_VEX_Integrated_Encoder_Modules

The Shaft Encoder Video - This video teaches students how shaft encoders work and how they are used to control distance.

The Motor and Sensors Setup, Forward for Distance 1 Video - This video teaches students how to configure the shaft encoders using the Motors and Setup wizard.

The Encoder, Forward for Distance 2 Video - This video teaches students why clearing the values in the encoder is important and introduces them to how "While Loops" work.



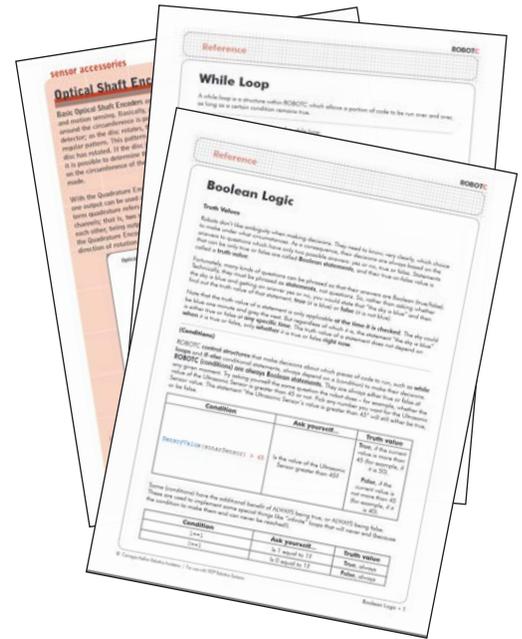
Movement/Shaft Encoders (continued)

Boolean Logic Part 1 & Part 2 Videos - Boolean Logic enables robots to make decisions. 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.

Boolean Logic PDF - Covers the same topics as the Boolean Logic videos, but in a print format.

Shaft Encoders PDF - Teaches students how shaft encoders work.

The While Loop Handout - Shows students the code that controls a while loop.



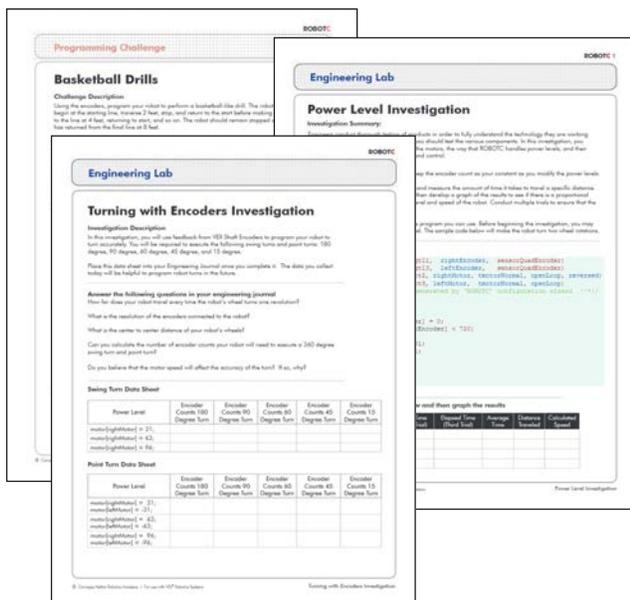
Additional Shaft Encoder Lesson Resources

The Sensor Debug Window Video - ROBOTC's debug window is one of its most powerful tools. This video introduces students to how they can see all of the robot's input and output values.

Shaft Encoders, Forward and Turning Video - Teaches students how to use encoders in path planning.

The Basketball Challenge - A programming challenge designed to give student's practice with the shaft encoders.

The Power Level and Turning with Encoder's Engineering Investigations - students use feedback from the "Debug" window to gather data to complete these two Engineering Investigations.



Movement/Shaft Encoders (continued)



Basketball Drills, Power Levels with Encoders, and Turning with Encoders are available as RVW Challenges.

Movement/Automated Straightening

The screenshot shows the ROBOTC curriculum navigation menu. The 'MOVEMENT' tab is selected and highlighted in orange. Below the menu, a list of lessons is displayed, with '5. Automated Straightening' highlighted in a red box. Below this lesson, a sub-menu lists four parts: '1. Automated Straightening Part 1', '2. Automated Straightening Part 2', '3. Values and Variables Part 1', and '4. Values and Variables Part 2'. An 'EXPAND ALL' button is visible above the lesson list.

The Automated Straightening Lesson Set

This lesson set continues to introduce students to programming concepts like if-else statements, variable types, and global variables. Students will get more practice using Boolean Logic as they create their own self-straightening algorithm.

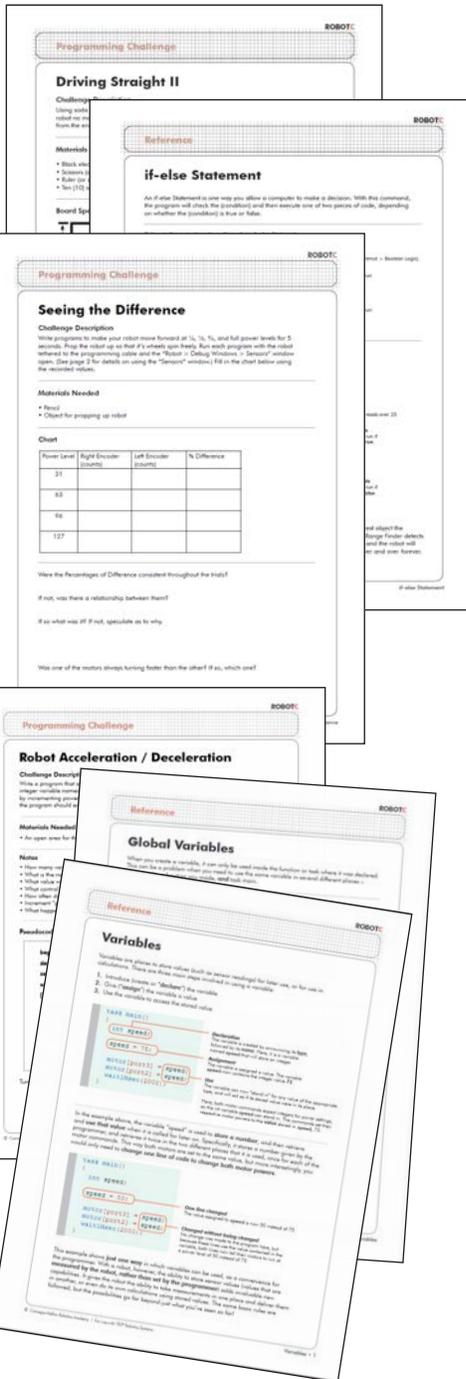
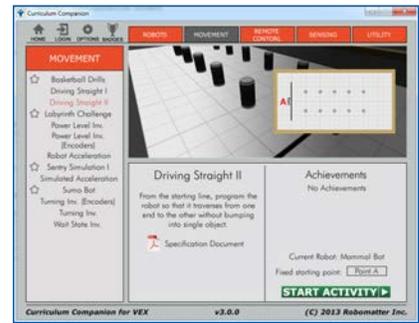
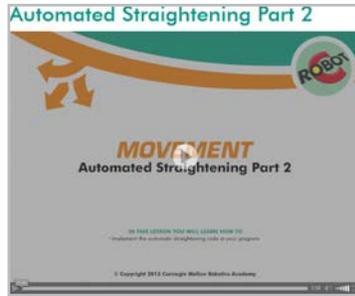
Automated Straightening Part 1 Video - This video teaches students will learn how to use feedback from encoders and conditional statements to develop an algorithm that allows the robot to self-correct its forward movements.

Automated Straightening Part 2 Video - This video teaches students how to implement the automated straightening algorithm on their robot.

if-else Statement Handout - This handout can be used as a study guide that shows students how the if-else Statement can be used with a while loop to help a robot make a decision.

Movement/Automated Straightening resources continued next page.

Movement/The Automated Straightening (continued)



The Driving Straight Programming Challenge - This challenge requires students to write a program that controls their robot to go perfectly straight. The challenge is also available using the RVW simulation software.

Seeing the Difference Challenge - This challenge requires students to use feedback from ROBOTC's real time debugger to track the robot's wheel movement.

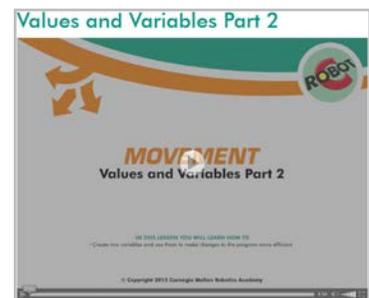
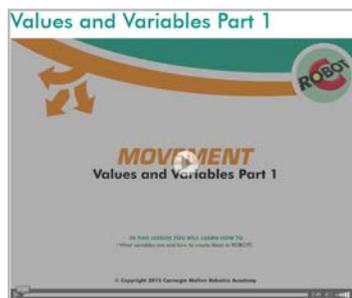
Boolean Logic Part 1 & Part 2 Videos - Boolean Logic enables robots to make decisions. The first video teaches students how conditional statements work. The second teaches them about logical operators. There is also a Boolean Logic 3 page PDF that complements the videos.

Variables and Values

Variables and Values Video Part 1 - This video introduces students to the power of variables. This lesson teaches students about how to specify variables and variable types..

Variables and Values Video Part 2 - This teaches students about variable names and types, how to initialize a variable, and how to use variables in their program.

Variable and Global Variables Reference Guide - This PDF can be used as a study guide and covers everything taught in the two variable videos.



Integrated Encoders

The screenshot shows the ROBOTC 66 interface with the 'MOVEMENT' section selected. The navigation bar at the top includes HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. The 'MOVEMENT' section is expanded, showing a list of lessons:

1. Challenge
2. Moving Forward
3. Speed and Direction
4. Shaft Encoder
5. Automated Straightening
6. Integrated Encoders

The '6. Integrated Encoders' lesson is expanded, showing a description and a list of sub-lessons:

Using the Shaft Encoders to control the distance the robot travels.

1. Forward for Distance IME
2. Principles of PID
3. Forward for Distance PID
4. Forward for Target Distance

The Integrated Encoders Lesson Set

This lesson set contains four videos designed to introduce students to the Integrated Motor Encoders and PID.

Forward for Distance IME Video - This video teaches students will learn how to use feedback from encoders and conditional statements to develop an algorithm that allows the robot to self-correct its forward movements.

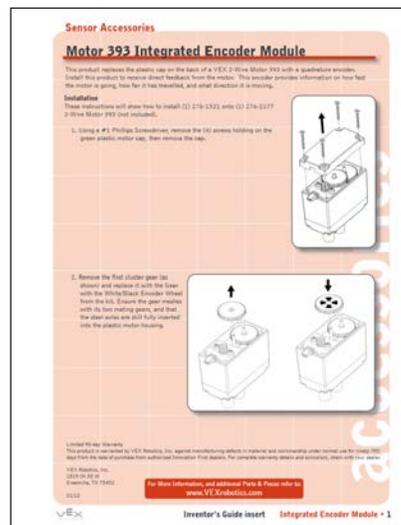
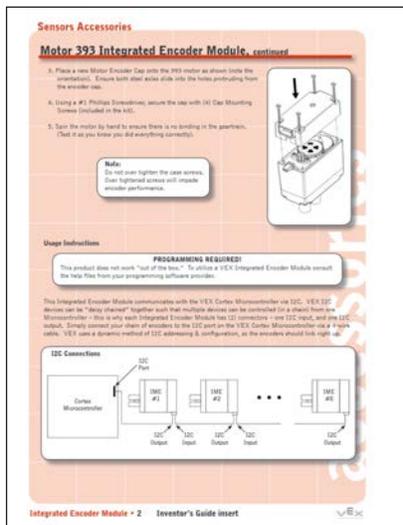
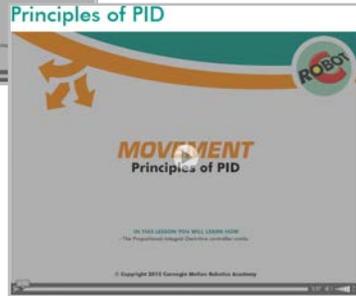
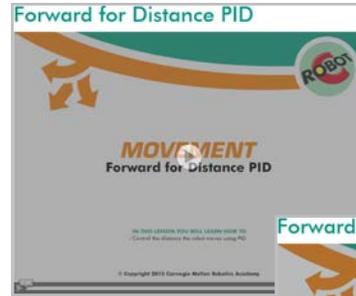
Principles of PID Video - This video teaches students how PID enabled robots can automate the ability to track how far each motor spins and make automated adjustments as the robot moves.

Forward for Distance PID Video - This video shows how to setup your motors programmatically to use PID and how to observe the values of the motors using the debug window.

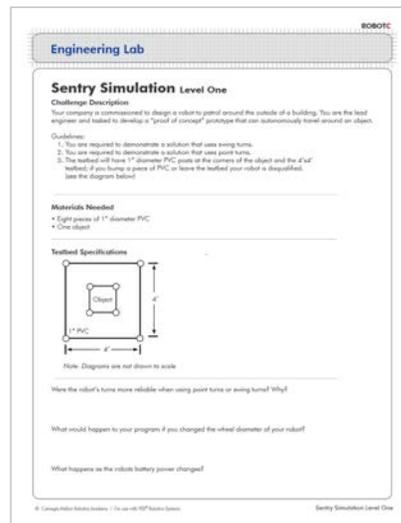
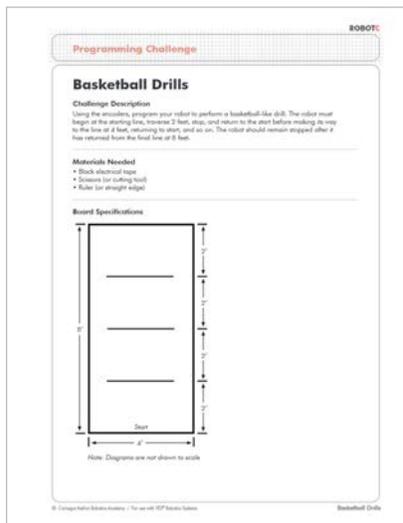
Forward for Target Distance PID Video - This video teaches how to use the `moveMotorTarget` and `getMotorEncoder` commands eliminating the robot drifting past the target location.

Integrated Encoder resources continued next page.

Integrated Encoders (continued)



The VEX Cortex Integrated Motor Encoders need to be installed. This lesson set includes a step-by-step set of instructions that you will need to install your IMEs.



Note: Any time a student needs to run code in the virtual world to follow along with a video, they can use the Utility tables (huge table, metric or imperial distance table, turn table).

Now that you know how to use IMEs these lessons are designed to provide practice.

Remote Control/Challenge Description

HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING ENGINEERING REFERENCE



REMOTE CONTROL

EXPAND ALL

- 1. Challenge
- 2. Joystick Mapping
- 3. Timers
- 4. Buttons

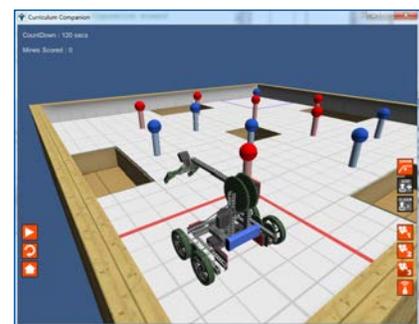
Note: Pictured at the right is the culminating challenge for the Remote Control Unit. Begin the unit by showing students the Minefield RC challenge. Students are not expected to know how to solve the challenge until they complete the Joystick Mapping, Timers, and Buttons Lessons.



Using Operator Input to Control Your Robot/The Minefield Challenge

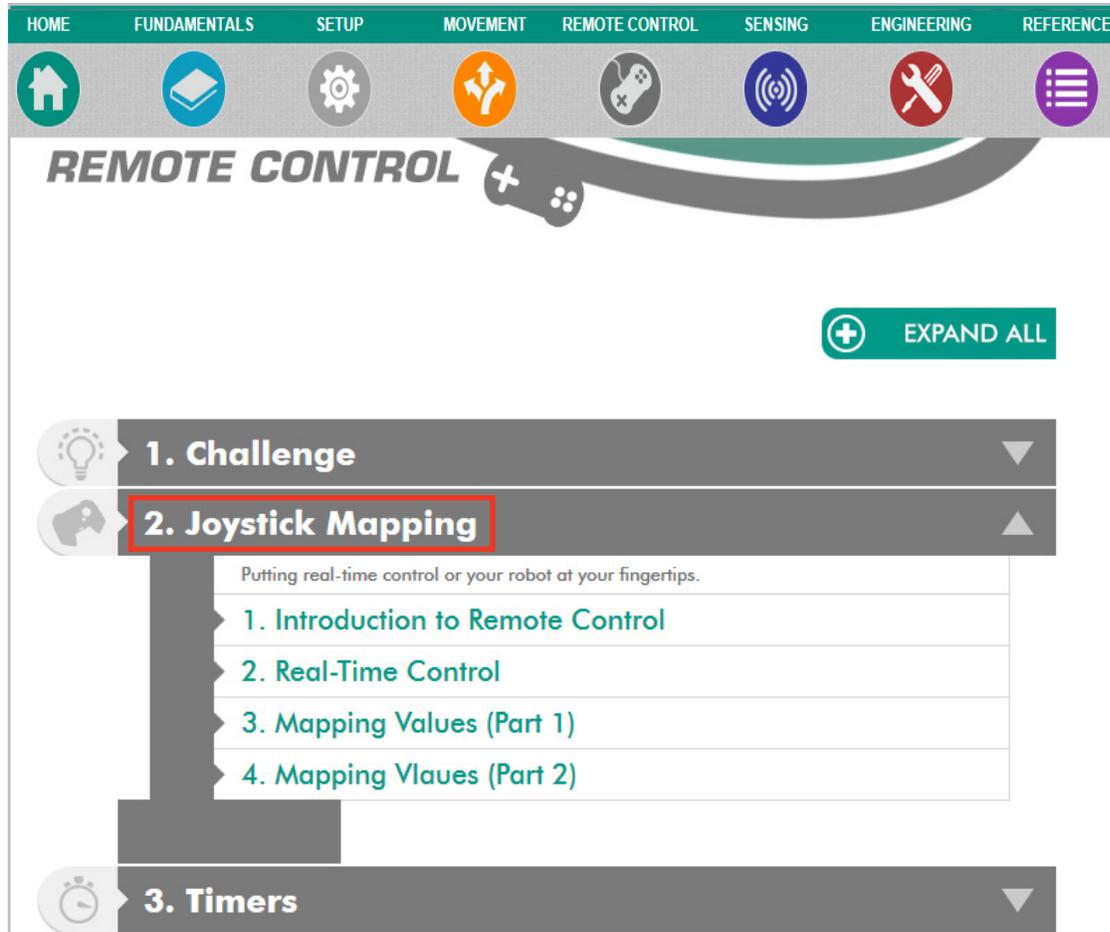
A powerful feature of the VEX Cortex is its ability to be driven using the VEXnet Remote Control. This ability proves to be indispensable in the competition environment. In this unit you will learn how to optimize and program your VEX remote control. You will learn to program buttons on the Joystick controller to enable robot behaviors.

The Minefield Challenge Resources - Remote Control Video Solution, Minefield Challenge PDF that describes the rules, and a Robot Virtual World simulation environment that allows students to develop and test code.



Note: the VEXnet Joystick does not communicate its values to the PC, only to a VEX Cortex, so it cannot be used to control the Virtual Robot. A USB Logitech Joystick is recommended for use with the virtual worlds. There are resources at the ROBOTC Wiki for using any USB joystick and also directly in ROBOTC's Help Documentation.

Remote Control/Joystick Mapping



Introduction to Remote Control Video - Students will learn how the VEXnet Remote Control works and be able to describe the range of values that the joystick provides. Additionally, they will learn which ROBOTC commands allow them to access the Remote Control.

Real Time Control Video - Students will learn how loops work and the difference between an infinite loop and a loop controlled by a conditional statement, they will also learn new ROBOTC reserved words that allow them to control the different remote controller channels.

VEXnet Joystick Calibration Guide - Initially, calibrating the Joystick may appear complicated to students. This is a four page guide with lots of pictures that takes them step-by-step through Joystick calibration.

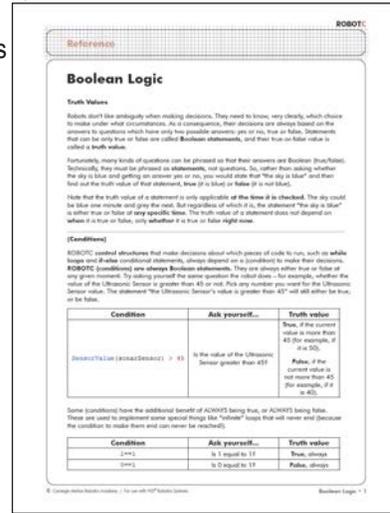
Joystick Mapping Resources are continued on the next page.



Remote Control/Joystick Mapping (continued)

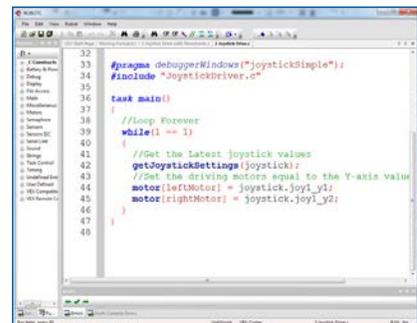
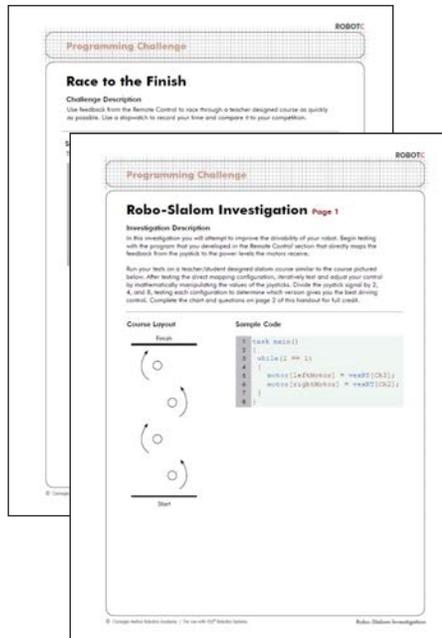
While Loops PDF - A simple example that shows how While Loops work.

Boolean Logic Part 1 & Part 2 Videos - The Boolean Logic videos are included in multiple lessons and are intended for review. These videos teach foundational principles that all programmers need to know. 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.



Mapping Values Videos Part 1 & 2 - These videos teach students how to map motor ports to the VEX joystick. By the end of the videos students will be able to change the motor speeds assigned by the remote control and will be able to optimize the joystick values to suit their needs.

Race to the Finish and Robo-Slalom Challenges - These are two relatively simple challenges that students will have fun with. They are designed to give the student practice optimizing the value and configuration of the VEX joystick.



ROBOTC Sample Programs - ROBOTC contains three sample programs in the Remote Control Folder, 2 Joystick Drive with Thresholds, 2 Joystick Drive, and Top Hat Drive. Students are encouraged to begin by opening the sample programs and modifying them.

Remote Control/Timers



REMOTE CONTROL

- 1. Challenge
- 2. Joystick Mapping
- 3. Timers
- 4. Buttons

Controlling behaviors with values collected over time.

- 1. Time and Timers
- 2. Using Timers



Time and Timers Video - In this video students will learn the difference between using the Wait1Msec command and using Timers to control behaviors, they will also learn how many Timers are available, how to reset Timers, and how to access the value of a timer programmatically.

The Using Timers Video - This video teaches students how to use the reserved words built into ROBOTC to control timers.

The Timers PDF Reference Guide - is a one page handout that can be used as a study guide for using Timers.

The Round-Up and Bull-In-The-Ring Challenges - Designed to give students the opportunity to apply Timers. Both challenges are available as Robot Virtual World Simulations.

Timers

Timers are very useful for performing a more complex behavior for a certain period of time. They allow you to set a delay (Wait) for the robot to wait before performing an action, which is fine for simple behaviors like moving forward. If calculations or other actions need to occur during the timed period, an with the loop tracking behavior below, a Timer must be used.

```

task main()
{
    while(!SensorTouch[Sensor1])
    {
        ClearTimer(T1);
        CreateTimer(T1, 3000);
        if(SensorTouch[Sensor1] == 1)
        {
            motor[port1] = 45;
            motor[port2] = 45;
        }
        else
        {
            motor[port1] = 0;
            motor[port2] = 0;
        }
    }
}
    
```

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

```
ClearTimer(T1name, number);
```

The VEX 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 measure the value of the timer by using `Time1[Sensor1]`, `Time2[Sensor1]`, or `Time3[Sensor1]` depending on whether you want the output to be in 1, 10, or 100 millisecond values. In the example above, you should see on the condition that we used time [T1]. The robot will back in line with the value of the timer to be less than 3 seconds. The program ends after 3 seconds.



Remote Control/Buttons

HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING ENGINEERING REFERENCE

REMOTE CONTROL EXPAND ALL

1. Challenge
2. Joystick Mapping
3. Timers
4. Buttons
 - Giving your robot the power to compete in the Minefield Challenge.
 - 1. Remote Control Buttons
 - 2. Remote Start
 - 3. Controlling the Arm Part 1
 - 4. Controlling the Arm Part 2
 - 5. Controlling the Arm Part 3

Buttons provide a very powerful input enabling students to control their robot's behaviors. This unit teaches students how to develop code that uses feedback from the remote control's buttons to control their robot's movements.

The Remote Control Buttons Video - This video teaches students the numbering systems for buttons on the VEX remote control. It also shows them how to access the value of a button, and the values that buttons send to the controller.

The Remote Start Video - This video teaches students how to develop competition ready code and how an idle loop can be used to control the start of a program.

The Controlling the Arm Video Part 1 - This video introduces students to programming an arm for the Minefield Challenge, how to use the Motors and Sensors Setup Wizard to rename motors, and how they can use an if statements to control the up and down motion for the arm.

Controlling the Arm Video Part 2 - This video teaches students how the else branch can be used to complete their arm control algorithm.

Controlling the Arm Video Part 3 - The final video shows students how to integrate a while loop and several if-else statements to control the robot's arm.

Additional Remote Control labs - The Remote Control Engineering Lab provides students with code and a set of activities designed to take them step-by-step through a remote control lab. Robo-Dunk, RoboWriter, and Turn Buttons are additional labs designed to provide a set of hands on activities to check a student's understanding of programming buttons.

Remote Control/Buttons (continued)

Introduction to Remote Control Buttons



Engineering Lab

Remote Control Buttons

In this tutorial you will:

1. Program the buttons on your remote control.
2. Identify the names and locations of all buttons on the VEXnet Remote Control.

Remote Control Overview

The VEXnet Remote Control is a very powerful tool that a programmer can use to use to achieve direct control of their robot. Each button can be programmed to control a specific behavior, for example: goLeft(), goRight(), leftTurn, openDrifter, closeDrifter, allowing endless options.

Joysticks: Each remote control has two joysticks. They are the round knobs that are labeled 1 + 2 and 3 + 4 on the picture on the left. To access the 3-axis of right joystick the command would be "wvRF(3)30000".

The joystick axis names are:

- CS1
- CS2
- CS3
- CS4

Note: ROBOTC has the capability of working with two remote controls at a time. Names for the second remote control are appended by #2. For example, to access the 3-axis of right joystick on the second remote control, the command would be "wvRF(2)30000".

Buttons: There are 12 programmable buttons on the remote control. The eight buttons on the front are broken into two groups of four, each having up, down, left, and right buttons. Two groups of up and down buttons make up the additional four buttons on the top of the remote control.

Accessing button values on ROBOTC is a very similar to accessing joystick values. The wvRF() command is still used, but now you use the letters "B" followed by the group number & button to, and finally the letter U, L, or R, depending on the button direction. For example, if you wanted to access the value of the down button on the front-left of the remote control you would use wvRF(3)B4U. For example, to access the value of the down button on the front-left of the remote control, the command would be "wvRF(3)B4U20000".

Note: Button names for the second remote control are appended by #2. For example, to access the value of the down button on the front-left of the remote control, the command would be "wvRF(2)B4U20000".

Controlling the Arm - Broken into 3 videos due to the complexity and length of the topic.



Button Labs - Available in PDF format and as a Robot Virtual World programming challenge.

Programming Challenge

RoboWriter

Challenge Description

Write a program that causes the robot to turn right when BnF is pressed, move straight forward when BnU is pressed, and stop when BnD is pressed. The joystick should have no effect on the robot's course before using your program and the Remote Control.

Materials Needed

- 1 VEXnet Joystick
- 1 Dry Erase Marker
- Sharp Pencil/Marker

Board Specifications

Note: Diagrams are not drawn to scale.

Programming Challenge

Robo-Dunk

Challenge Description

Program your robot to use the buttons on the back of the Remote Control to pick up a ball or half court and drop it into the scoring area on the far end. Remote Control must be used to accomplish this task.

Programming Challenge

Turn Buttons

Challenge Description

Write a program that causes the robot to turn right when BnR is pressed, move straight forward when BnU is pressed, and stop when BnD is pressed. The joystick should have no effect on the robot's course before using your program and the Remote Control.

Materials Needed

- Block electrical tape
- Scissors (or cutting tool)

Engineering Lab

Remote Control Buttons

In this tutorial you will:

1. Program the buttons on your remote control.
2. Identify the names and locations of all buttons on the VEXnet Remote Control.

Remote Control Overview

The VEXnet Remote Control is a very powerful tool that a programmer can use to use to achieve direct control of their robot. Each button can be programmed to control a specific behavior, for example: goLeft(), goRight(), leftTurn, openDrifter, closeDrifter, allowing endless options.

Joysticks: Each remote control has two joysticks. They are the round knobs that are labeled 1 + 2 and 3 + 4 on the picture on the left. To access the 3-axis of right joystick the command would be "wvRF(3)30000".

The joystick axis names are:

- CS1
- CS2
- CS3
- CS4

Note: ROBOTC has the capability of working with two remote controls at a time. Names for the second remote control are appended by #2. For example, to access the 3-axis of right joystick on the second remote control, the command would be "wvRF(2)30000".

Buttons: There are 12 programmable buttons on the remote control. The eight buttons on the front are broken into two groups of four, each having up, down, left, and right buttons. Two groups of up and down buttons make up the additional four buttons on the top of the remote control.

Accessing button values on ROBOTC is a very similar to accessing joystick values. The wvRF() command is still used, but now you use the letters "B" followed by the group number & button to, and finally the letter U, L, or R, depending on the button direction. For example, if you wanted to access the value of the down button on the front-left of the remote control you would use wvRF(3)B4U. For example, to access the value of the down button on the front-left of the remote control, the command would be "wvRF(3)B4U20000".

Note: Button names for the second remote control are appended by #2. For example, to access the value of the down button on the front-left of the remote control, the command would be "wvRF(2)B4U20000".

Note: the VEXnet Joystick does not communicate its values to the PC, only to a VEX Cortex, so it cannot be used to control the Virtual Robot. A USB Logitech Joystick is recommended for use with the virtual worlds. There are resources at the ROBOTC Wiki for using any USB joystick and also directly in ROBOTC's Help Documentation.

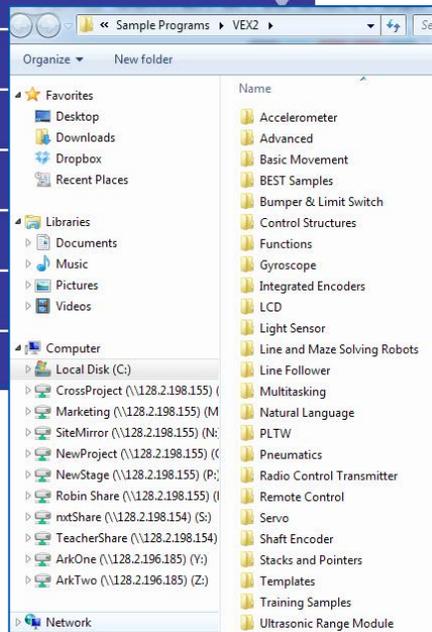
Sensing



Robots separate themselves from simpler machines through their ability to detect and respond to their surroundings. In this section, you will learn how to take advantage of the multiple sensors available to the VEX Cortex.

EXPAND ALL

- 1. Challenge
- 2. Limiting the Arm
- 3. Behaviors and Functions
- 4. Forward until Near
- 5. Line Tracking
- 6. Turn for Angle
- 7. Using the LCD



Sensing/Additional Resources

Since the release of the VEX Cortex robot VEX Robotics has added the Gyro, Accelerometer, and Integrated Motor Encoders. There are four places that you can find help with new sensors or ROBOTC features.

1. **Sample programs** - whenever ROBOTC adds new sensor functionality you will find sample programs that demonstrate how the sensor works.
2. **ROBOTC Help Documentation**- Developers also document new features at the wiki. http://www.robotc.net/wiki/VEX2_Sensors_Overview
3. **The ROBOTC Blog** - Developers like to show the community what is new and blog about the new features. <http://www.robotc.net/blog/>
4. **The ROBOTC Forum** - ROBOTC has an active community that helps others with their code. <http://www.robotc.net/forums>

Sensing/The Grand Challenge

HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING ENGINEERING REFERENCE

SENSING

EXPAND ALL

1. Challenge

Robots separate themselves from simpler machines through their ability to detect and respond to their surroundings. The Grand Challenge will put both your robot and your programming skills to the test.

1. Grand Challenge

2. Limiting the Arm

3. Behaviors and Functions

The Grand Challenge

This challenge is a teacher designed course that reinforces behavior-based programming and the engineering process. The challenge course is teacher-designed course and not revealed to students until the day of the competition. Before the competition, students are provided with a list of conditions and situations to prepare for (width of the robot's path, 30 degree ramps, line following, obstacle detection, etc). On the day of the competition, the student's programming knowledge and preparation are put to the test as they work to traverse the course in a limited amount of time. The robot that makes the most progress without stalling out or deviating from the course wins!

Example Robot Behaviors/Functions - The list below are example behaviors and functions that students might use to solve the Grand Challenge.

- Moving Straight using Encoders
- Accurately Move a Specific Distance
- Move Forward Until the Light Sensor Sees a Dark Line
- Program using Remote Control
- Precise Turning using Encoder Feedback
- Precise Turning using Gyro Sensor Feedback
- Programming Remote Controller Buttons to Precise Turning using Encoder or Gyro Sensor Feedback
- Obstacle Detection using Feedback from Sonar or Touch Sensors (autonomous)
- Track a Line
- Ability to Push Object
- Ability to Detect and Count Lines
- Ability to Display Characters to the LCD Remote Screen

Grand Challenge resources are found on the next page.

Grand Challenge Problem Resources

Physical Robot Resources

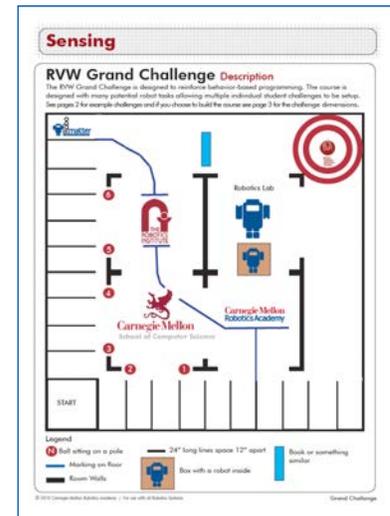
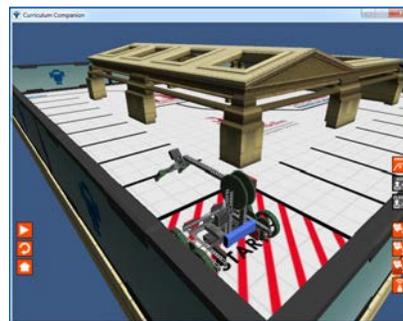
The Grand Challenge Design Specification - A PDF that describes the challenge.

Grand Challenge Example Solution Video - This solution video is an example of a course. The course that you use will be designed by you and your students.

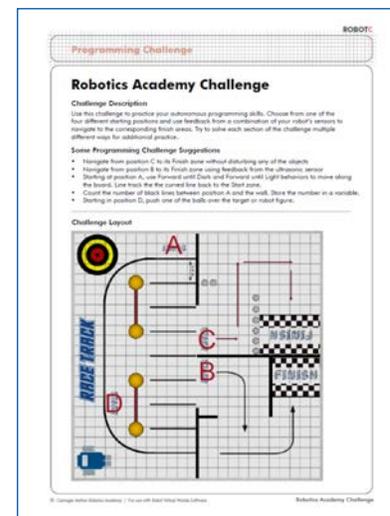
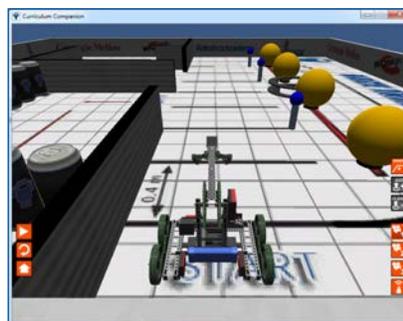


Robot Virtual World Resources

The Robot Virtual World Grand Challenge Design Specification - This world is designed so that it can be solved many ways allowing the teacher to assign student challenges based on ability. Below, at the left is a picture of the home screen for the RVW Grand Challenge, it is found in the Utility Tables section, in the center is a screen shot of the RVW, and at the right is the RVW Grand Challenge programming description PDF.



The Robot Virtual World Robotics Academy Grand Challenge Design Specification - This world is also designed so that it can be solved many ways allowing the teacher to assign student challenges based on ability. This world is a little easier than the RVW Grand Challenge above. Below, at the left is a picture of the home screen for the RVW Grand Challenge, it is found in the Utility Tables section, in the center is a screen shot of the RVW, and at the right is the Robotics Academy RVW Grand Challenge programming description PDF.



Sensing/Limiting The Arm

The screenshot shows the ROBOTC software interface. At the top, there is a navigation bar with tabs for HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, SENSING, ENGINEERING, and REFERENCE. Below the navigation bar, the 'SENSING' section is active, indicated by a large blue wave graphic and the word 'SENSING' in bold. A green button with a plus sign and the text 'EXPAND ALL' is located in the upper right of the sensing area. Below this, there are three main menu items: '1. Challenge', '2. Limiting the Arm', and '3. Behaviors and Functions'. The '2. Limiting the Arm' item is highlighted with a red rectangular box. A sub-menu is expanded from '2. Limiting the Arm', showing three sub-items: '1. Configuring Sensors', '2. Limiting the Arm Part 1', and '3. Limiting the Arm Part 2'. A descriptive text for the '2. Limiting the Arm' item reads: 'Using the Limit Switch and Potentiometer to effectively control the robotic arm.'

Limiting the Arm Resources

The Limiting the Arm lesson set teaches students how to configure and integrate sensors into the lift-arm mechanism on the robot. The Lesson Set has the following resources:

The Configuring Sensors Video - The video explains how to configure sensors using ROBOTC's Motors and Sensors Setup wizard and reviews the rules for naming sensors.

The Sensors Reference Guide PDF - A 12 page reference guide that describes the Analog and Digital ports found on VEX Controllers and describes how touch sensors work.

Servo Motors Reference Guide PDF - A 2 page reference guide that shows students how to control servo motors.

Limiting the Arm Part 1 Video - This video explains how the touch sensor works as a limit switch to control how far the motor can move. Students will also learn how to apply the AND Logical Operator in their conditional statement enabling them to check two conditions at the same time.

if-else Statement Reference Guide PDF - Provides students with a review of how the if-else Statement works.

Switch Cases Reference Guide PDF - Provides students with working code that shows how a Switch Case works.

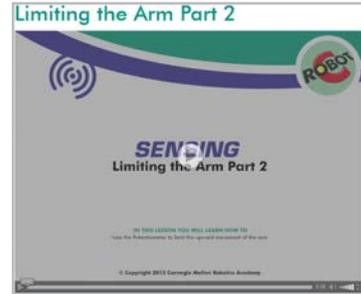
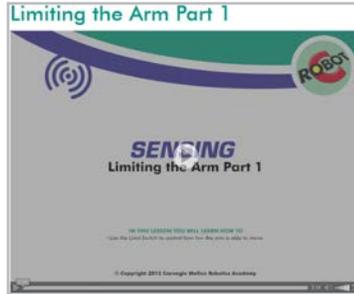
Limiting the Arm Part 2 Video - This video teaches about potentiometers and how they work. Students will learn how to integrate the potentiometer into the mechanical arm mechanism using ROBOTC's debug window.

Potentiometer Reference Guides - The reference guides are designed to be used as study guides or teaching tools and explain how the sensor works.

Lifting the Arm resources are continued on the next page.

Sensing/Limiting the Arm Resources (continued)

Instructional Videos



PDF Reference Guides

Sensor

Table of Contents:

- Introduction to the Sensor Subsystem 5.2
- Concepts to Understand 5.3
- Subsystem Interactions 5.11

Investor's Guide 5 • 1

Reference

Servo Module Overview

A Servo Module (or Servo Hub) rotates its shaft to a set angular position, between 0 and 180 degrees. Once its position has been set in a ROBOTC program, the Servo Module will continuously draw power to maintain that position until another is specified. Servo Modules can be plugged into any of the RCXSER ports on ROBOTC.

The only difference in appearance between the VEX Servo and Motor Modules are their shafts on the back, but the two should not be confused. When a Motor Module is set equal to a value, it uses that value as a power setting and shaft spinning the shaft in continuous motion. When a Servo Module is set equal to a value, however, it uses that value to rotate its shaft to a position and hold it there.

Servo Modules are typically and appropriately found in robotic grippers and arms because of their small range of motion, ability to be set to a specific position, and ability to hold that position. They can be set to rotate ranging from -137 to 137, with -137 being fully rotated one way and 137 fully rotated the other.

Mounted Servo Module: Here, the Servo Module is mounted on a robot's top deck. It is a good idea to label and wire up specific ports.

Servo Module

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 Statement.

```

if (condition)
    // code to execute if condition is true
else
    // code to execute if condition is false
    
```

Below is an example program containing an if-else Statement.

```

task main()
{
    while (true)
    {
        if (SensorValue[SensorA] > 25)
        {
            motor[port4] = 127;
            motor[port2] = 127;
        }
        else
        {
            motor[port4] = 0;
            motor[port2] = 0;
        }
    }
}
    
```

This if-else Statement tells the robot to run both motors at full power if the sensor detects the Ultrasonic Range Finder detects a more than 25 inches away. If the Ultrasonic Range Finder detects an object closer than 25 inches, then the "else" portion of the code will be run and the robot will stop moving. The while (true) loop makes the if-else statement run over and over forever.

if-else Statement

Reference

Switch Case

The switch case command is a decision-making statement which chooses commands to run from a list of separate "cases". A single "switch" value is selected and evaluated, and different sets of code are run based on which "case" the value matches.

Below is the pseudocode outline of a switch case Statement.

```

switch (switch_value)
{
    case value_1:
        // code to run for value_1
    case value_2:
        // code to run for value_2
    case value_3:
        // code to run for value_3
    default:
        // code to run if no other cases match
}
    
```

Switch Case • 1

Reference

Potentiometers Overview

The Potentiometer is used to measure the angular position of the shaft or shaft passed through its center. The center of the sensor can rotate roughly 240 degrees and outputs ranging from 0 to 1023 in the V5S RCX and 0 to 65535 in the V5X Cortex.

The Potentiometer can be attached to the robot using the mounting hole surrounding the center of the sensor. The shaft provides feedback for the orientation of the Potentiometer, allowing the full range of motion to be utilized more easily.

When mounted on the rotating shaft of a moving portion of the robot, such as an arm or gripper, the Potentiometer provides precise feedback regarding the angular position. This sensor data can then be used for accurate control of the robot.

Mounted Potentiometer: Here, the Potentiometer is mounted on a robot's top deck. It is a good idea to label and wire up specific ports.

CAUTION: When mounting the Potentiometer on your robot, ensure that the range of motion of the rotating shaft does not exceed that of the sensor. Failure to do so may result in damage to your robot and the Potentiometer.

Check if Size: If the range of motion of the rotating shaft is greater than the range of motion of the Potentiometer, the sensor will not be able to measure the position of the shaft.

Potentiometer

sensor accessories

Potentiometer Kit

The Vex Potentiometer will keep things "on level". Use this sensor to get an analog measurement of angular position. This measurement can help to understand the position of robot arms, or other mechanisms. To efficiently utilize this sensor, users are required to use the Vex Programming Kit.

YOU MUST HAVE A PROGRAMMING KIT TO USE THIS SENSOR!

Kit includes:

- (2) Potentiometer Assembly
- (4) 9-12 x 1/4" Long Mounting Screws
- (4) 12 x 1/2" Long Mounting Screws

The Potentiometer is designed with a "0 hole" in the center. This hole should slide easily over the Vex square shafts. The Potentiometer also includes (2) "arcs" which are 1/2" from the center hole; these arcs are used for mounting the Potentiometer to the robot structure.

The mounting arcs allow for 90-degree adjustment to the Potentiometer position. Since the Potentiometer has limited travel, it is important to ensure that the shaft that is being measured by the Potentiometer does not rotate more than 240-degrees. The Potentiometer can only rotate mechanically about 245-degrees ± 5 and can only measure electrically 250-degrees ± 20. The adjustment arcs allow the Potentiometer's "range of motion" to be repositioned to match the shaft's range of motion. To measure the motion of something which moves more than 230-degrees, try gearing down the shaft's motion to a secondary shaft (this secondary shaft will move less distance) and then measure this secondary shaft.

Potentiometer Kit • 1

Sensing/Limiting the Arm Resources (continued)

Physical Robot Programming Challenges

Programming Challenge

Wall Follower

Challenge Description
Program your robot to follow along a wall with the aid of a limit switch. The limit switch should be placed along the left side of the robot, allowing it to make contact with the wall. You will need to modify your standard robot build to complete this challenge.

Materials Needed
• A flat wall, at least 8" long

Board Specifications

Note: Diagrams are not drawn to scale.

Programming Challenge

Robotic Mouse

Challenge Description
Design, program, test and troubleshoot a robot that follows along multiple walls (like a mouse) using feedback from either two limit switches on a limit switch and a bumper switch. You will need to modify your standard robot build to complete this challenge.

Materials Needed
• An area with at least 4 walls

Board Specifications

Note: Diagrams are not drawn to scale.

Programming Challenge

Robocci

Challenge Description
Using a limit switch or ultrasonic sensor, program your robot to move forward until it makes contact with a table (or without moving it or knocking it over). It should then stop the amount of time it took to make contact into an integer variable named "timeToTime".

Keep your robot connected to the computer over VEKnet or USB to observe the value stored in "timeToTime" using the Robot > Debugger Windows > Global Variables window. (See page 2 for details on using the "Global Variables" window.)

Materials Needed
• Block electrical tape
• 1 table top

Board Specifications

Note: Diagrams are not drawn to scale.

Programming Challenge

Quick Tap 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 "totalCount". 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" into the variable named "totalCount".

Keep your robot connected to the computer over VEKnet or USB to observe the value stored in "totalCount" 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 in the value of the variable "pressCount". This is going to be problematic if your robot is stuck on the sensor.
- Increment "pressCount" using the code: "pressCount = pressCount + 1".
- Begin by following pseudocode that describes in detail what you need 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 "totalCount"
  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 "totalCount" equal to "pressCount"
end program
    
```

Turn the pseudocode example into ROBOTC code and test it.

Programming Challenge

Addition and Subtraction

Challenge Description
This challenge requires that you have two limit switches connected to your robot.

Create a program with an integer variable named "pressCount". Whenever the first limit switch is pressed, "pressCount" should be incremented by 1. Whenever the second limit switch is pressed, "pressCount" should be decremented by 1.

Keep your robot connected to the computer over VEKnet or USB to observe the value stored in "pressCount" using the Robot > Debugger Windows > Global Variables window. (See page 2 for details on using the "Global Variables" window.)

Notes

- "pressCount" can be incremented using the code: "pressCount = pressCount + 1".
- "pressCount" can be decremented using the code: "pressCount = pressCount - 1".
- Make sure that you initialize "pressCount" at the beginning of your program to 0.
- Each "press" should result in a change of only +/- 1.

Robot Virtual World Programming Challenges



Sensing/Behaviors and Functions

The screenshot shows the ROBOTC software interface. At the top, there is a navigation bar with six tabs: HOME, FUNDAMENTALS, SETUP, MOVEMENT, REMOTE CONTROL, and SENSING. Below the tabs are icons for each category. The SENSING tab is selected, and the word "SENSING" is displayed in large blue letters. A blue curved banner with a wireless signal icon is positioned below the tabs. To the right of the banner is a green button with a plus sign and the text "EXPAND ALL". Below the banner is a list of menu items:

- 1. Challenge
- 2. Limiting the Arm
- 3. Behaviors and Functions
 - 1. Behaviors and Functions 1
 - 2. Behaviors and Functions 2
 - 3. Passing Parameters Part 1
 - 4. Passing Parameters Part 2

This lesson set will teach students to build their own behaviors by creating functions. They will also learn to pass parameters that make their functions more portable.

The Behaviors and Functions Part 1 Video - This video introduces students to the power of functions enabling them to make their programs shorter and easier to read.

The Behaviors and Functions Part 2 Video - In this lesson students will learn to transform their code into functions. They will learn how to declare a function and what a parameter is.

The Shaft Encoders Reference Guide - This PDF explains how Shaft Encoders work.

VEX Integrated Encoders Module - Video lessons, challenges, and helper pages.

Optimizing Code and Incorporating Functions Challenges - Two challenges that give students the chance to revisit old code that they wrote and make the code easier to read by incorporating functions into their code.

Passing Parameters Part 1 Video - In this video students learn how to pass parameters allowing them to write more powerful functions, they will also learn the rules to naming and specifying parameters.

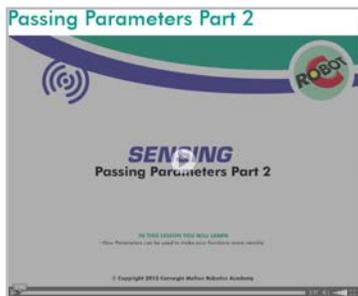
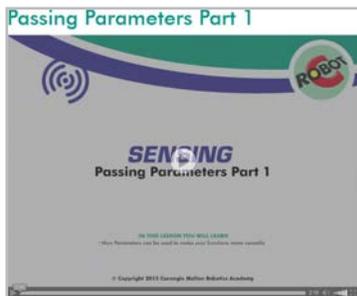
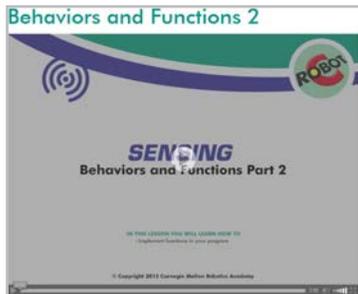
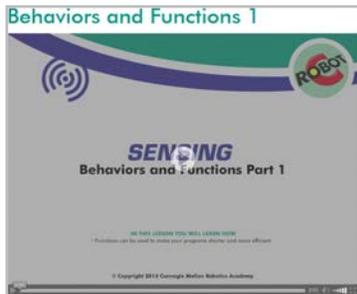
Passing Parameters Part 2 Video - In this lesson students learn how to implement the parameter including the type of data the parameter will pass and the name of the parameter.

Function Challenges - The Real World Values, RoboDunk2, Seeing the Difference, and Robot Acceleration Programming Challenge enable students to revisit code that they've previously written and optimize it by incorporating functions and parameters.

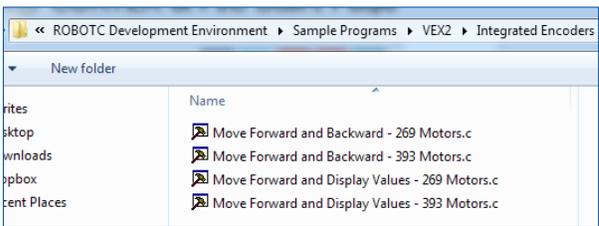
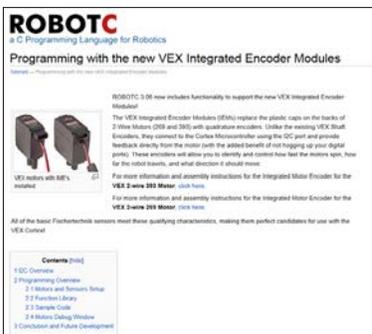
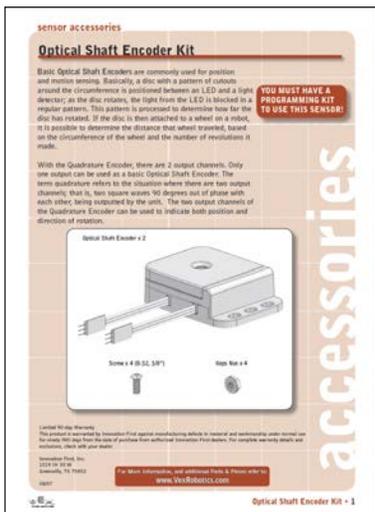
Additional resources for the Sensing/Behaviors and Functions lesson set can be found on the next page.

Sensing/Behaviors and Functions Resources (Continued)

Instructional Videos



PDF Reference Guides



Commented Integrated Encoder sample programs are found in the current ROBOTC Software build.

Additional resources are available on the next page.

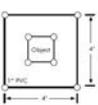
Sensing/Behaviors and Functions Resources (Continued)

Physical Robot Programming Challenges

Programming Challenge

Optimizing Code

Challenge Description
The sample program below is made up of the same behaviors, used over and over. Write a program that uses structures such as functions and loops to minimize the number of lines of code necessary to perform the same functionality as the Sample Code below. To verify that you haven't changed the actual behavior of the robot, download and run both your program and the sample program on your robot. Note that you can only use one sensor to verify the robot; the robot successfully moves within the course.



Sample Program

```

1 task main()
2 {
3   motor[motorA] = 40;
4   motor[motorB] = 40;
5   wait1Motor(2000);
6
7   motor[motorA] = -40;
8   motor[motorB] = 40;
9   wait1Motor(1000);
10
11  motor[motorA] = 40;
12  motor[motorB] = -40;
13  wait1Motor(1000);
14
15  motor[motorA] = 40;
16  motor[motorB] = 40;
17  wait1Motor(2000);
18
19  motor[motorA] = -40;
20  motor[motorB] = 40;
21  wait1Motor(1000);
22
23  motor[motorA] = 40;
24  motor[motorB] = -40;
25  wait1Motor(1000);
26
27  motor[motorA] = 40;
28  motor[motorB] = 40;
29  wait1Motor(2000);
30
31  motor[motorA] = -40;
32  motor[motorB] = 40;
33  wait1Motor(1000);
34
35  motor[motorA] = 40;
36  motor[motorB] = -40;
37  wait1Motor(1000);
38
39  motor[motorA] = 40;
40  motor[motorB] = 40;
41  wait1Motor(2000);
42
43  motor[motorA] = -40;
44  motor[motorB] = 40;
45  wait1Motor(1000);
46
47  motor[motorA] = 40;
48  motor[motorB] = -40;
49  wait1Motor(1000);
50
51  motor[motorA] = 40;
52  motor[motorB] = 40;
53  wait1Motor(2000);
54
55  motor[motorA] = -40;
56  motor[motorB] = 40;
57  wait1Motor(1000);
58
59  motor[motorA] = 40;
60  motor[motorB] = -40;
61  wait1Motor(1000);
62
63  motor[motorA] = 40;
64  motor[motorB] = 40;
65  wait1Motor(2000);
66
67  motor[motorA] = -40;
68  motor[motorB] = 40;
69  wait1Motor(1000);
70
71  motor[motorA] = 40;
72  motor[motorB] = -40;
73  wait1Motor(1000);
74
75  motor[motorA] = 40;
76  motor[motorB] = 40;
77  wait1Motor(2000);
78
79  motor[motorA] = -40;
80  motor[motorB] = 40;
81  wait1Motor(1000);
82
83  motor[motorA] = 40;
84  motor[motorB] = -40;
85  wait1Motor(1000);
86
87  motor[motorA] = 40;
88  motor[motorB] = 40;
89  wait1Motor(2000);
90
91  motor[motorA] = -40;
92  motor[motorB] = 40;
93  wait1Motor(1000);
94
95  motor[motorA] = 40;
96  motor[motorB] = -40;
97  wait1Motor(1000);
98
99  motor[motorA] = 40;
100 motor[motorB] = 40;
101 wait1Motor(2000);
102
103 motor[motorA] = -40;
104 motor[motorB] = 40;
105 wait1Motor(1000);
106
107 motor[motorA] = 40;
108 motor[motorB] = -40;
109 wait1Motor(1000);
110
111 motor[motorA] = 40;
112 motor[motorB] = 40;
113 wait1Motor(2000);
114
115 motor[motorA] = -40;
116 motor[motorB] = 40;
117 wait1Motor(1000);
118
119 motor[motorA] = 40;
120 motor[motorB] = -40;
121 wait1Motor(1000);
122
123 motor[motorA] = 40;
124 motor[motorB] = 40;
125 wait1Motor(2000);
126
127 motor[motorA] = -40;
128 motor[motorB] = 40;
129 wait1Motor(1000);
130
131 motor[motorA] = 40;
132 motor[motorB] = -40;
133 wait1Motor(1000);
134
135 motor[motorA] = 40;
136 motor[motorB] = 40;
137 wait1Motor(2000);
138
139 motor[motorA] = -40;
140 motor[motorB] = 40;
141 wait1Motor(1000);
142
143 motor[motorA] = 40;
144 motor[motorB] = -40;
145 wait1Motor(1000);
146
147 motor[motorA] = 40;
148 motor[motorB] = 40;
149 wait1Motor(2000);
150
151 motor[motorA] = -40;
152 motor[motorB] = 40;
153 wait1Motor(1000);
154
155 motor[motorA] = 40;
156 motor[motorB] = -40;
157 wait1Motor(1000);
158
159 motor[motorA] = 40;
160 motor[motorB] = 40;
161 wait1Motor(2000);
162
163 motor[motorA] = -40;
164 motor[motorB] = 40;
165 wait1Motor(1000);
166
167 motor[motorA] = 40;
168 motor[motorB] = -40;
169 wait1Motor(1000);
170
171 motor[motorA] = 40;
172 motor[motorB] = 40;
173 wait1Motor(2000);
174
175 motor[motorA] = -40;
176 motor[motorB] = 40;
177 wait1Motor(1000);
178
179 motor[motorA] = 40;
180 motor[motorB] = -40;
181 wait1Motor(1000);
182
183 motor[motorA] = 40;
184 motor[motorB] = 40;
185 wait1Motor(2000);
186
187 motor[motorA] = -40;
188 motor[motorB] = 40;
189 wait1Motor(1000);
190
191 motor[motorA] = 40;
192 motor[motorB] = -40;
193 wait1Motor(1000);
194
195 motor[motorA] = 40;
196 motor[motorB] = 40;
197 wait1Motor(2000);
198
199 motor[motorA] = -40;
200 motor[motorB] = 40;
201 wait1Motor(1000);
202
203 motor[motorA] = 40;
204 motor[motorB] = -40;
205 wait1Motor(1000);
206
207 motor[motorA] = 40;
208 motor[motorB] = 40;
209 wait1Motor(2000);
210
211 motor[motorA] = -40;
212 motor[motorB] = 40;
213 wait1Motor(1000);
214
215 motor[motorA] = 40;
216 motor[motorB] = -40;
217 wait1Motor(1000);
218
219 motor[motorA] = 40;
220 motor[motorB] = 40;
221 wait1Motor(2000);
222
223 motor[motorA] = -40;
224 motor[motorB] = 40;
225 wait1Motor(1000);
226
227 motor[motorA] = 40;
228 motor[motorB] = -40;
229 wait1Motor(1000);
230
231 motor[motorA] = 40;
232 motor[motorB] = 40;
233 wait1Motor(2000);
234
235 motor[motorA] = -40;
236 motor[motorB] = 40;
237 wait1Motor(1000);
238
239 motor[motorA] = 40;
240 motor[motorB] = -40;
241 wait1Motor(1000);
242
243 motor[motorA] = 40;
244 motor[motorB] = 40;
245 wait1Motor(2000);
246
247 motor[motorA] = -40;
248 motor[motorB] = 40;
249 wait1Motor(1000);
250
251 motor[motorA] = 40;
252 motor[motorB] = -40;
253 wait1Motor(1000);
254
255 motor[motorA] = 40;
256 motor[motorB] = 40;
257 wait1Motor(2000);
258
259 motor[motorA] = -40;
260 motor[motorB] = 40;
261 wait1Motor(1000);
262
263 motor[motorA] = 40;
264 motor[motorB] = -40;
265 wait1Motor(1000);
266
267 motor[motorA] = 40;
268 motor[motorB] = 40;
269 wait1Motor(2000);
270
271 motor[motorA] = -40;
272 motor[motorB] = 40;
273 wait1Motor(1000);
274
275 motor[motorA] = 40;
276 motor[motorB] = -40;
277 wait1Motor(1000);
278
279 motor[motorA] = 40;
280 motor[motorB] = 40;
281 wait1Motor(2000);
282
283 motor[motorA] = -40;
284 motor[motorB] = 40;
285 wait1Motor(1000);
286
287 motor[motorA] = 40;
288 motor[motorB] = -40;
289 wait1Motor(1000);
290
291 motor[motorA] = 40;
292 motor[motorB] = 40;
293 wait1Motor(2000);
294
295 motor[motorA] = -40;
296 motor[motorB] = 40;
297 wait1Motor(1000);
298
299 motor[motorA] = 40;
300 motor[motorB] = -40;
301 wait1Motor(1000);
302
303 motor[motorA] = 40;
304 motor[motorB] = 40;
305 wait1Motor(2000);
306
307 motor[motorA] = -40;
308 motor[motorB] = 40;
309 wait1Motor(1000);
310
311 motor[motorA] = 40;
312 motor[motorB] = -40;
313 wait1Motor(1000);
314
315 motor[motorA] = 40;
316 motor[motorB] = 40;
317 wait1Motor(2000);
318
319 motor[motorA] = -40;
320 motor[motorB] = 40;
321 wait1Motor(1000);
322
323 motor[motorA] = 40;
324 motor[motorB] = -40;
325 wait1Motor(1000);
326
327 motor[motorA] = 40;
328 motor[motorB] = 40;
329 wait1Motor(2000);
330
331 motor[motorA] = -40;
332 motor[motorB] = 40;
333 wait1Motor(1000);
334
335 motor[motorA] = 40;
336 motor[motorB] = -40;
337 wait1Motor(1000);
338
339 motor[motorA] = 40;
340 motor[motorB] = 40;
341 wait1Motor(2000);
342
343 motor[motorA] = -40;
344 motor[motorB] = 40;
345 wait1Motor(1000);
346
347 motor[motorA] = 40;
348 motor[motorB] = -40;
349 wait1Motor(1000);
350
351 motor[motorA] = 40;
352 motor[motorB] = 40;
353 wait1Motor(2000);
354
355 motor[motorA] = -40;
356 motor[motorB] = 40;
357 wait1Motor(1000);
358
359 motor[motorA] = 40;
360 motor[motorB] = -40;
361 wait1Motor(1000);
362
363 motor[motorA] = 40;
364 motor[motorB] = 40;
365 wait1Motor(2000);
366
367 motor[motorA] = -40;
368 motor[motorB] = 40;
369 wait1Motor(1000);
370
371 motor[motorA] = 40;
372 motor[motorB] = -40;
373 wait1Motor(1000);
374
375 motor[motorA] = 40;
376 motor[motorB] = 40;
377 wait1Motor(2000);
378
379 motor[motorA] = -40;
380 motor[motorB] = 40;
381 wait1Motor(1000);
382
383 motor[motorA] = 40;
384 motor[motorB] = -40;
385 wait1Motor(1000);
386
387 motor[motorA] = 40;
388 motor[motorB] = 40;
389 wait1Motor(2000);
390
391 motor[motorA] = -40;
392 motor[motorB] = 40;
393 wait1Motor(1000);
394
395 motor[motorA] = 40;
396 motor[motorB] = -40;
397 wait1Motor(1000);
398
399 motor[motorA] = 40;
400 motor[motorB] = 40;
401 wait1Motor(2000);
402
403 motor[motorA] = -40;
404 motor[motorB] = 40;
405 wait1Motor(1000);
406
407 motor[motorA] = 40;
408 motor[motorB] = -40;
409 wait1Motor(1000);
410
411 motor[motorA] = 40;
412 motor[motorB] = 40;
413 wait1Motor(2000);
414
415 motor[motorA] = -40;
416 motor[motorB] = 40;
417 wait1Motor(1000);
418
419 motor[motorA] = 40;
420 motor[motorB] = -40;
421 wait1Motor(1000);
422
423 motor[motorA] = 40;
424 motor[motorB] = 40;
425 wait1Motor(2000);
426
427 motor[motorA] = -40;
428 motor[motorB] = 40;
429 wait1Motor(1000);
430
431 motor[motorA] = 40;
432 motor[motorB] = -40;
433 wait1Motor(1000);
434
435 motor[motorA] = 40;
436 motor[motorB] = 40;
437 wait1Motor(2000);
438
439 motor[motorA] = -40;
440 motor[motorB] = 40;
441 wait1Motor(1000);
442
443 motor[motorA] = 40;
444 motor[motorB] = -40;
445 wait1Motor(1000);
446
447 motor[motorA] = 40;
448 motor[motorB] = 40;
449 wait1Motor(2000);
450
451 motor[motorA] = -40;
452 motor[motorB] = 40;
453 wait1Motor(1000);
454
455 motor[motorA] = 40;
456 motor[motorB] = -40;
457 wait1Motor(1000);
458
459 motor[motorA] = 40;
460 motor[motorB] = 40;
461 wait1Motor(2000);
462
463 motor[motorA] = -40;
464 motor[motorB] = 40;
465 wait1Motor(1000);
466
467 motor[motorA] = 40;
468 motor[motorB] = -40;
469 wait1Motor(1000);
470
471 motor[motorA] = 40;
472 motor[motorB] = 40;
473 wait1Motor(2000);
474
475 motor[motorA] = -40;
476 motor[motorB] = 40;
477 wait1Motor(1000);
478
479 motor[motorA] = 40;
480 motor[motorB] = -40;
481 wait1Motor(1000);
482
483 motor[motorA] = 40;
484 motor[motorB] = 40;
485 wait1Motor(2000);
486
487 motor[motorA] = -40;
488 motor[motorB] = 40;
489 wait1Motor(1000);
490
491 motor[motorA] = 40;
492 motor[motorB] = -40;
493 wait1Motor(1000);
494
495 motor[motorA] = 40;
496 motor[motorB] = 40;
497 wait1Motor(2000);
498
499 motor[motorA] = -40;
500 motor[motorB] = 40;
501 wait1Motor(1000);
502
503 motor[motorA] = 40;
504 motor[motorB] = -40;
505 wait1Motor(1000);
506
507 motor[motorA] = 40;
508 motor[motorB] = 40;
509 wait1Motor(2000);
510
511 motor[motorA] = -40;
512 motor[motorB] = 40;
513 wait1Motor(1000);
514
515 motor[motorA] = 40;
516 motor[motorB] = -40;
517 wait1Motor(1000);
518
519 motor[motorA] = 40;
520 motor[motorB] = 40;
521 wait1Motor(2000);
522
523 motor[motorA] = -40;
524 motor[motorB] = 40;
525 wait1Motor(1000);
526
527 motor[motorA] = 40;
528 motor[motorB] = -40;
529 wait1Motor(1000);
530
531 motor[motorA] = 40;
532 motor[motorB] = 40;
533 wait1Motor(2000);
534
535 motor[motorA] = -40;
536 motor[motorB] = 40;
537 wait1Motor(1000);
538
539 motor[motorA] = 40;
540 motor[motorB] = -40;
541 wait1Motor(1000);
542
543 motor[motorA] = 40;
544 motor[motorB] = 40;
545 wait1Motor(2000);
546
547 motor[motorA] = -40;
548 motor[motorB] = 40;
549 wait1Motor(1000);
550
551 motor[motorA] = 40;
552 motor[motorB] = -40;
553 wait1Motor(1000);
554
555 motor[motorA] = 40;
556 motor[motorB] = 40;
557 wait1Motor(2000);
558
559 motor[motorA] = -40;
560 motor[motorB] = 40;
561 wait1Motor(1000);
562
563 motor[motorA] = 40;
564 motor[motorB] = -40;
565 wait1Motor(1000);
566
567 motor[motorA] = 40;
568 motor[motorB] = 40;
569 wait1Motor(2000);
570
571 motor[motorA] = -40;
572 motor[motorB] = 40;
573 wait1Motor(1000);
574
575 motor[motorA] = 40;
576 motor[motorB] = -40;
577 wait1Motor(1000);
578
579 motor[motorA] = 40;
580 motor[motorB] = 40;
581 wait1Motor(2000);
582
583 motor[motorA] = -40;
584 motor[motorB] = 40;
585 wait1Motor(1000);
586
587 motor[motorA] = 40;
588 motor[motorB] = -40;
589 wait1Motor(1000);
590
591 motor[motorA] = 40;
592 motor[motorB] = 40;
593 wait1Motor(2000);
594
595 motor[motorA] = -40;
596 motor[motorB] = 40;
597 wait1Motor(1000);
598
599 motor[motorA] = 40;
600 motor[motorB] = -40;
601 wait1Motor(1000);
602
603 motor[motorA] = 40;
604 motor[motorB] = 40;
605 wait1Motor(2000);
606
607 motor[motorA] = -40;
608 motor[motorB] = 40;
609 wait1Motor(1000);
610
611 motor[motorA] = 40;
612 motor[motorB] = -40;
613 wait1Motor(1000);
614
615 motor[motorA] = 40;
616 motor[motorB] = 40;
617 wait1Motor(2000);
618
619 motor[motorA] = -40;
620 motor[motorB] = 40;
621 wait1Motor(1000);
622
623 motor[motorA] = 40;
624 motor[motorB] = -40;
625 wait1Motor(1000);
626
627 motor[motorA] = 40;
628 motor[motorB] = 40;
629 wait1Motor(2000);
630
631 motor[motorA] = -40;
632 motor[motorB] = 40;
633 wait1Motor(1000);
634
635 motor[motorA] = 40;
636 motor[motorB] = -40;
637 wait1Motor(1000);
638
639 motor[motorA] = 40;
640 motor[motorB] = 40;
641 wait1Motor(2000);
642
643 motor[motorA] = -40;
644 motor[motorB] = 40;
645 wait1Motor(1000);
646
647 motor[motorA] = 40;
648 motor[motorB] = -40;
649 wait1Motor(1000);
650
651 motor[motorA] = 40;
652 motor[motorB] = 40;
653 wait1Motor(2000);
654
655 motor[motorA] = -40;
656 motor[motorB] = 40;
657 wait1Motor(1000);
658
659 motor[motorA] = 40;
660 motor[motorB] = -40;
661 wait1Motor(1000);
662
663 motor[motorA] = 40;
664 motor[motorB] = 40;
665 wait1Motor(2000);
666
667 motor[motorA] = -40;
668 motor[motorB] = 40;
669 wait1Motor(1000);
670
671 motor[motorA] = 40;
672 motor[motorB] = -40;
673 wait1Motor(1000);
674
675 motor[motorA] = 40;
676 motor[motorB] = 40;
677 wait1Motor(2000);
678
679 motor[motorA] = -40;
680 motor[motorB] = 40;
681 wait1Motor(1000);
682
683 motor[motorA] = 40;
684 motor[motorB] = -40;
685 wait1Motor(1000);
686
687 motor[motorA] = 40;
688 motor[motorB] = 40;
689 wait1Motor(2000);
690
691 motor[motorA] = -40;
692 motor[motorB] = 40;
693 wait1Motor(1000);
694
695 motor[motorA] = 40;
696 motor[motorB] = -40;
697 wait1Motor(1000);
698
699 motor[motorA] = 40;
700 motor[motorB] = 40;
701 wait1Motor(2000);
702
703 motor[motorA] = -40;
704 motor[motorB] = 40;
705 wait1Motor(1000);
706
707 motor[motorA] = 40;
708 motor[motorB] = -40;
709 wait1Motor(1000);
710
711 motor[motorA] = 40;
712 motor[motorB] = 40;
713 wait1Motor(2000);
714
715 motor[motorA] = -40;
716 motor[motorB] = 40;
717 wait1Motor(1000);
718
719 motor[motorA] = 40;
720 motor[motorB] = -40;
721 wait1Motor(1000);
722
723 motor[motorA] = 40;
724 motor[motorB] = 40;
725 wait1Motor(2000);
726
727 motor[motorA] = -40;
728 motor[motorB] = 40;
729 wait1Motor(1000);
730
731 motor[motorA] = 40;
732 motor[motorB] = -40;
733 wait1Motor(1000);
734
735 motor[motorA] = 40;
736 motor[motorB] = 40;
737 wait1Motor(2000);
738
739 motor[motorA] = -40;
740 motor[motorB] = 40;
741 wait1Motor(1000);
742
743 motor[motorA] = 40;
744 motor[motorB] = -40;
745 wait1Motor(1000);
746
747 motor[motorA] = 40;
748 motor[motorB] = 40;
749 wait1Motor(2000);
750
751 motor[motorA] = -40;
752 motor[motorB] = 40;
753 wait1Motor(1000);
754
755 motor[motorA] = 40;
756 motor[motorB] = -40;
757 wait1Motor(1000);
758
759 motor[motorA] = 40;
760 motor[motorB] = 40;
761 wait1Motor(2000);
762
763 motor[motorA] = -40;
764 motor[motorB] = 40;
765 wait1Motor(1000);
766
767 motor[motorA] = 40;
768 motor[motorB] = -40;
769 wait1Motor(1000);
770
771 motor[motorA] = 40;
772 motor[motorB] = 40;
773 wait1Motor(2000);
774
775 motor[motorA] = -40;
776 motor[motorB] = 40;
777 wait1Motor(1000);
778
779 motor[motorA] = 40;
780 motor[motorB] = -40;
781 wait1Motor(1000);
782
783 motor[motorA] = 40;
784 motor[motorB] = 40;
785 wait1Motor(2000);
786
787 motor[motorA] = -40;
788 motor[motorB] = 40;
789 wait1Motor(1000);
790
791 motor[motorA] = 40;
792 motor[motorB] = -40;
793 wait1Motor(1000);
794
795 motor[motorA] = 40;
796 motor[motorB] = 40;
797 wait1Motor(2000);
798
799 motor[motorA] = -40;
800 motor[motorB] = 40;
801 wait1Motor(1000);
802
803 motor[motorA] = 40;
804 motor[motorB] = -40;
805 wait1Motor(1000);
806
807 motor[motorA] = 40;
808 motor[motorB] = 40;
809 wait1Motor(2000);
810
811 motor[motorA] = -40;
812 motor[motorB] = 40;
813 wait1Motor(1000);
814
815 motor[motorA] = 40;
816 motor[motorB] = -40;
817 wait1Motor(1000);
818
819 motor[motorA] = 40;
820 motor[motorB] = 40;
821 wait1Motor(2000);
822
823 motor[motorA] = -40;
824 motor[motorB] = 40;
825 wait1Motor(1000);
826
827 motor[motorA] = 40;
828 motor[motorB] = -40;
829 wait1Motor(1000);
830
831 motor[motorA] = 40;
832 motor[motorB] = 40;
833 wait1Motor(2000);
834
835 motor[motorA] = -40;
836 motor[motorB] = 40;
837 wait1Motor(1000);
838
839 motor[motorA] = 40;
840 motor[motorB] = -40;
841 wait1Motor(1000);
842
843 motor[motorA] = 40;
844 motor[motorB] = 40;
845 wait1Motor(2000);
846
847 motor[motorA] = -40;
848 motor[motorB] = 40;
849 wait1Motor(1000);
850
851 motor[motorA] = 40;
852 motor[motorB] = -40;
853 wait1Motor(1000);
854
855 motor[motorA] = 40;
856 motor[motorB] = 40;
857 wait1Motor(2000);
858
859 motor[motorA] = -40;
860 motor[motorB] = 40;
861 wait1Motor(1000);
862
863 motor[motorA] = 40;
864 motor[motorB] = -40;
865 wait1Motor(1000);
866
867 motor[motorA] = 40;
868 motor[motorB] = 40;
869 wait1Motor(2000);
870
871 motor[motorA] = -40;
872 motor[motorB] = 40;
873 wait1Motor(1000);
874
875 motor[motorA] = 40;
876 motor[motorB] = -40;
877 wait1Motor(1000);
878
879 motor[motorA] = 40;
880 motor[motorB] = 40;
881 wait1Motor(2000);
882
883 motor[motorA] = -40;
884 motor[motorB] = 40;
885 wait1Motor(1000);
886
887 motor[motorA] = 40;
888 motor[motorB] = -40;
889 wait1Motor(1000);
890
891 motor[motorA] = 40;
892 motor[motorB] = 40;
893 wait1Motor(2000);
894
895 motor[motorA] = -40;
896 motor[motorB] = 40;
897 wait1Motor(1000);
898
899 motor[motorA] = 40;
900 motor[motorB] = -40;
901 wait1Motor(1000);
902
903 motor[motorA] = 40;
904 motor[motorB] = 40;
905 wait1Motor(2000);
906
907 motor[motorA] = -40;
908 motor[motorB] = 40;
909 wait1Motor(1000);
910
911 motor[motorA] = 40;
912 motor[motorB] = -40;
913 wait1Motor(1000);
914
915 motor[motorA] = 40;
916 motor[motorB] = 40;
917 wait1Motor(2000);
918
919 motor[motorA] = -40;
920 motor[motorB] = 40;
921 wait1Motor(1000);
922
923 motor[motorA] = 40;
924 motor[motorB] = -40;
925 wait1Motor(1000);
926
927 motor[motorA] = 40;
928 motor[motorB] = 40;
929 wait1Motor(2000);
930
931 motor[motorA] = -40;
932 motor[motorB] = 40;
933 wait1Motor(1000);
934
935 motor[motorA] = 40;
936 motor[motorB] = -40;
937 wait1Motor(1000);
938
939 motor[motorA] = 40;
940 motor[motorB] = 40;
941 wait1Motor(2000);
942
943 motor[motorA] = -40;
944 motor[motorB] = 40;
945 wait1Motor(1000);
946
947 motor[motorA] = 40;
948 motor[motorB] = -40;
949 wait1Motor(1000);
950
951 motor[motorA] = 40;
952 motor[motorB] = 40;
953 wait1Motor(2000);
954
955 motor[motorA] = -40;
956 motor[motorB] = 40;
957 wait1Motor(1000);
958
959 motor[motorA] = 40;
960 motor[motorB] = -40;
961 wait1Motor(1000);
962
963 motor[motorA] = 40;
964 motor[motorB] = 40;
965 wait1Motor(2000);
966
967 motor[motorA] = -40;
968 motor[motorB] = 40;
969 wait1Motor(1000);
970
971 motor[motorA] = 40;
972 motor[motorB] = -40;
973 wait1Motor(1000);
974
975 motor[motorA] = 40;
976 motor[motorB] = 40;
977 wait1Motor(2000);
978
979 motor[motorA] = -40;
980 motor[motorB] = 40;
981 wait1Motor(1000);
982
983 motor[motorA] = 40;
984 motor[motorB] = -40;
985 wait1Motor(1000);
986
987 motor[motorA] = 40;
988 motor[motorB] = 40;
989 wait1Motor(2000);
990
991 motor[motorA] = -40;
992 motor[motorB] = 40;
993 wait1Motor(1000);
994
995 motor[motorA] = 40;
996 motor[motorB] = -40;
997 wait1Motor(1000);
998
999 motor[motorA] = 40;
1000 motor[motorB] = 40;
1001 wait1Motor(2000);
1002
1003 motor[motorA] = -40;
1004 motor[motorB] = 40;
1005 wait1Motor(1000);
1006
1007 motor[motorA] = 40;
1008 motor[motorB] = -40;
1009 wait1Motor(1000);
1010
1011 motor[motorA] = 40;
1012 motor[motorB] = 40;
1013 wait1Motor(2000);
1014
1015 motor[motorA] = -40;
1016 motor[motorB] = 40;
1017 wait1Motor(1000);
1018
1019 motor[motorA] = 40;
1020 motor[motorB] = -40;
1021 wait1Motor(1000);
1022
1023 motor[motorA] = 40;
1024 motor[motorB] = 40;
1025 wait1Motor(2000);
1026
1027 motor[motorA] = -40;
1028 motor[motorB] = 40;
1029 wait1Motor(1000);
1030
1031 motor[motorA] = 40;
1032 motor[motorB] = -40;
1033 wait1Motor(1000);
1034
1035 motor[motorA] = 40;
1036 motor[motorB] = 40;
1037 wait1Motor(2000);
1038
1039 motor[motorA] = -40;
1040 motor[motorB] = 40;
1041 wait1Motor(1000);
1042
1043 motor[motorA] = 40;
1044 motor[motorB] = -40;
1045 wait1Motor(1000);
1046
1047 motor[motorA] = 40;
1048 motor[motorB] = 40;
1049 wait1Motor(2000);
1050
1051 motor[motorA] = -40;
1052 motor[motorB] = 40;
1053 wait1Motor(1000);
1054
1055 motor[motorA] = 40;
1056 motor[motorB] = -40;
1057 wait1Motor(1000);
1058
1059 motor[motorA] = 40;
1060 motor[motorB] = 40;
1061 wait1Motor(2000);
1062
1063 motor[motorA] = -40;
1064 motor[motorB] = 40;
1065 wait1Motor(1000);
1066
1067 motor[motorA] = 40
```

Sensing/Forward Until Near

The screenshot shows the ROBOTC software interface with the 'SENSING' menu expanded. The top navigation bar includes 'HOME', 'FUNDAMENTALS', 'SETUP', 'MOVEMENT', 'REMOTE CONTROL', and 'SENSING'. The 'SENSING' menu is open, displaying a list of lessons: '1. Challenge', '2. Limiting the Arm', '3. Behaviors and Functions', and '4. Forward until Near'. The '4. Forward until Near' lesson is selected, and its sub-lessons are visible: '1. The Ultrasonic Rangefinder', '2. Forward until Near', '3. Straight until Near', and '4. Straight until Near (Fine Tuning)'. A description for the selected lesson reads: 'Moving near an object or obstacle using the Ultrasonic Rangefinder.'

This lesson set will teach students how to program the VEX Ultrasonic Rangefinder Sensor

The Ultrasonic Rangefinder Video - This video teaches students about the Ultrasonic Rangefinder.

The Ultrasonic Rangefinder Reference Guide - This guide explains how the Ultrasonic Rangefinder works.

The Forward Until Near Video - Shows students how the code for the Forward Until Near behavior works with the ultrasonic rangefinder including: using the Motors and Sensors wizard, the while loop, and the related code.

The Thresholds Reference Guide - A reference guide that explains what a threshold value is.

The Straight Until Near Video - This lesson combines the Forward Until Near behavior with the Automated Straightening behavior. They will also review how encoders work and why you need to clear them before use.

The Straight Until Near Fine Tuning Video - In this lesson the student will optimize their program using the debugging window so that it moves to the object and stops when it hits the object.

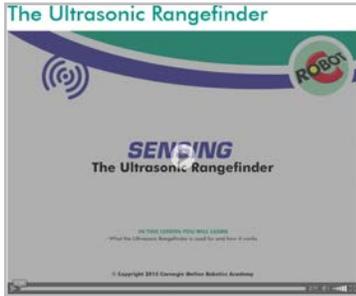
Forward Until Near Programming Challenges - This lesson set contains four increasingly difficult programming challenges that use the Ultrasonic Rangefinder: Sentry Simulation Two, The Speed of Sound, Sentry Simulation Level Three, and the Sonic Scanner programming challenges.

Robot Virtual World Programming Challenges - The RVW simulation software has the Sentry Simulation Level Two, Sentry Simulation Level Three, Speed of Sound, and the Sonic Scanner programming challenges in simulation.

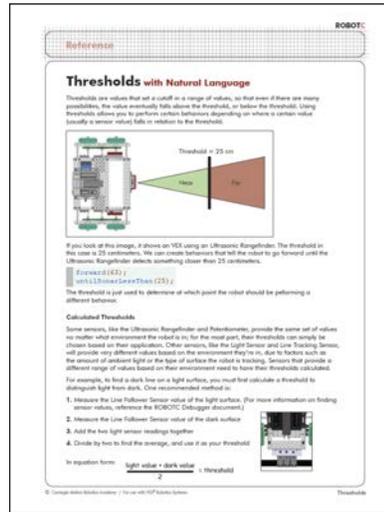
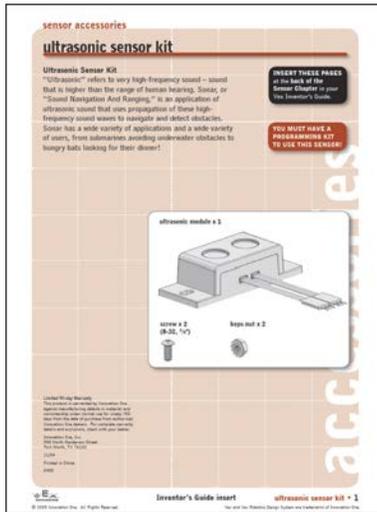
Additional resources for the Sensing/Behaviors and Functions lesson set can be found on the next page.

Sensing/Forward Until Near (continued)

Instructional Videos



PDF Reference Guides



Additional resources are available on the next page.

Sensing/Forward Until Near (continued)

Physical Robot Programming Challenges

Engineering Lab

Sentry Simulation Level Two

Challenge Description
Your competency level one Sentry Simulation was a big success. In the level two simulation the robot must recognize obstacles in its path and stop until the obstacle is removed. When the obstacle is removed, the robot needs to continue to patrol around the arena.

Questions:

1. The robot will patrol around the perimeter until an obstacle appears in its path.
2. It must stop when it sees the obstacle and then resume when the obstacle is removed.
3. The field will have 1" diameter PVC posts at the corners of the center object and the 4" x 8" field. If you bring a piece of PVC to leave the fielded your robot is disqualified. (see the diagram below)

Materials Needed

- Twelve pieces of 1" diameter PVC
- Three objects

Field Specifications

Note: Diagrams are not drawn to scale.

Describe how you solved the problem. Be specific. Use another sheet of paper to describe your solution. Place this in your Engineering Journal when finished.

Programming Challenge

The Speed of Sound

Challenge Description, part 1
Write a program that uses feedback from the Ultrasonic Rangefinder to control your robot's speed as it approaches an object. As the robot drives closer to a detectable object it should slowly decrease and slowly come to a complete stop to avoid contact with that object. Try several mathematical formulas to optimize your robot's deceleration speed.

Materials Needed

- A flat, open area for the robot to drive with a barrier at one end.

Notes

- Use the value in "SensorValue[sonarSensor]" to control the power level of the robot's motors.
- Try several mathematical operations to see which value gives you the best deceleration. Examples: `motor[RightMotor] = sensorValue[sonarSensor] * 3`, `motor[RightMotor] = sensorValue[sonarSensor] * 17`, or `motor[RightMotor] = sensorValue[sonarSensor] / 2`, `motor[LeftMotor] = sensorValue[sonarSensor] / 2`

For each method that you tried, how close did your robot actually get to objects and surfaces before it stopped driving?

For each method that you tried, did the motors actually stop at that point, or was power still being sent to the motors?

Describe in detail the effect that the different mathematical formulas had on the robot's deceleration. Is this what you expected would happen?

Engineering Lab

Sentry Simulation Level Three

Challenge Description
You completed the level one and level two Sentry Simulation programming labs. In Level Three, your robot must recognize obstacles in its path and stop until the obstacle is removed. As soon as the robot sees the obstacle, it must send a signal to base (your computer). When the obstacle is removed, the robot needs to continue to patrol around the arena.

Questions:

1. When an obstacle appears on the robot's path it must send a signal back to base. Consider changing the value of a variable, which can be displayed on the ROBOTC Global Variables display window.
2. It must stop when it sees the obstacle and then resume when the obstacle is removed.
3. The field will have 1" diameter PVC posts at the corners of the center object and the 4" x 8" field. If you bring a piece of PVC to leave the fielded your robot is disqualified. (see the diagram below)

Materials Needed

- Twelve pieces of 1" diameter PVC
- A circle of objects for obstacles
- A USB connection between the PC and robot

Field Specifications

Note: Diagrams are not drawn to scale.

Describe how you solved the problem. Be specific. Use another sheet of paper to describe your solution. Place this in your Engineering Journal when finished.

Programming Challenge

Sonic Scanner

Challenge Description
Program your robot to slowly spin in place (about 1/2 inch) if no obstacle is returning global object sensor. If 18 inches or less of an Ultrasonic Rangefinder. Once the robot has detected the object, it should stop spinning and move towards the object. The robot should stop when it is within 3 inches of the object.

Run this program on your robot multiple times, making sure to stop the distance between the robot and the object. Use your observations to answer the questions that follow.

Materials Needed

- A flat, open area for the robot to spin
- 1 Detectable object (balls and pins can work)

Did your robot ever fail to recognize the object? If so, how far away was the object?

Show down the initial spinning of your robot and run several more trials. How did this affect the performance of your robot?

Change your program so that the robot will spin for an object up to 48 inches away. Run several more trials. How reliable is the Ultrasonic Rangefinder at further distances?

Robot Virtual World Programming Challenges

Sensing/Line Tracking

HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING

SENSING

EXPAND ALL

1. Challenge
2. Limiting the Arm
3. Behaviors and Functions
4. Forward until Near
5. Line Tracking
 1. The Line Tracking Sensors
 2. Calculating Thresholds
 3. Basic Line Tracking
 4. Line Tracking for Distance
 5. Optimized Line Tracking

The SwerveBot Building Instructions - The Swervebot tracks lines well and can be used for this challenge.

The Line Follower Kit Reference Guide - This PDF shows students how the light sensor kit works including the values the light sensor sees and the ports that it connects to.

The Line Tracking Calculating Thresholds Video - This video teaches students how to use ROBOTC's debugger to calculate threshold values for the light sensor.

The Thresholds Reference Guide - This reference guide explains what a threshold value is and how to calculate it.

The Variables Reference Guide - This reference guide explains what variables are and how to use them.

The Basic Line Tracking Video - This video teaches students how to build a line tracking behavior.

Line Tracking for Distance Video - This video teaches students how to use a combination of encoder values and the Line Track kit to track a line for a specific distance.

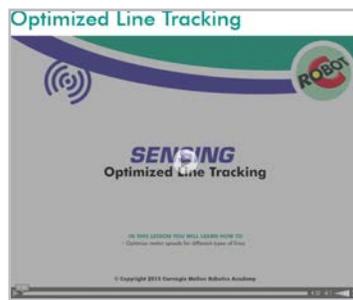
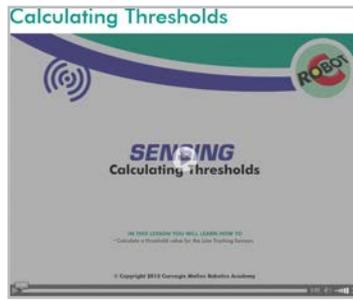
The Optimized Line Tracking Video - Line tracking can be slow, this video shows how you can modify the motor powers to significantly increase the speed your robot tracks a line.

Programming Challenges - This unit has the Forward Until Dark, TableBot, Robo500, Robo Slalom Level 2, Minefield Traversal and Robo Slalom Level three programming challenges to give students practice programming.

Additional resources for the Line Tracking lesson set can be found on the next page.

Sensing/Line Tracking (continued)

Instructional Videos



Reference

Variables

Variables are places to store values (such as sensor readings) for later use, or for use in calculations. There are three main steps involved in using a variable:

1. Introduce (create or "declare") the variable.
2. Give ("assign") the variable a value.
3. Use the variable to access the stored value.

```

1. int speed;
2. int speed = 75;
3. motor[port3] = speed;
   wait(1000);
   
```

Declaration: The variable is created by announcing its type followed by its name. Here, it is a variable named `speed` and it stores an integer.

Assignment: An assigned value, the variable `speed` is given the value of 75.

Use: The variable can now "hold" the value of the assignment and will act as if its stored value were in its place.

Example: The code above causes the motor to turn at the set variable speed for 1000 ms. The motor will stop after 1000 ms and will continue to the next line of code.

In the example above, the variable "speed" is used to store a number, and then returns and use that value when it is called for later use. Specifically, it stores a number given by the programmer, and returns it twice in the two different places that it is used, once for each of the motor commands. This way both motors are set to the same value, but more importantly you would only need to change one line of code to change both motor powers.

```

1. int speed;
2. int speed = 50;
3. motor[port3] = speed;
   wait(1000);
   
```

How this changed: The value assigned to speed is now 50 instead of 75.

Changed without being changed: The change was made to the program text, but because that line was not used, the value of the variable was not changed. The value of the variable is still 50.

This example shows just one way in which variables can be used, as a convenience for the programmer. With a value, however, the ability to store sensor values allows them to be measured by the robot, rather than set by the programmer. This is useful in applications where you need to give the robot the ability to take measurements in one place and deliver them in another, or even do the same calculations using stored values. The same basic rules are followed, but the possibilities go far beyond just what you've seen so far!

© Carnegie Mellon Robotics Academy / For use with VEX Robotics Systems | Variables | 1

PDF Reference Guides

sensor accessories

line follower kit

Line Follower Kit
A line follower consists of an infrared light sensor and an infrared LED. It works by illuminating a surface with infrared radiation and, based on its intensity, determines the reflectivity of the surface in question. Light-colored surfaces will reflect more light than dark surfaces, resulting in their appearing brighter to the sensor. This allows the sensor to detect a dark line on a pale surface, or a pale line on a dark surface.

You can use a line follower to help your robot navigate along a marked path, or in any other application involving discerning the boundary between two high-contrast surfaces. A typical application uses three line follower sensors, such that the middle sensor is over the line your robot is following.

Use follower x 3

mounting bar x 1

screws x 3 (8-32, 1/2")

brackets x 3

INVERT THESE PAGES
Place this page in your VEX Cortex's Guide.

YOU MUST HAVE A PROGRAMMER SET TO USE THIS SENSOR!

Inventor's Guide insert | **Line Follower Kit • 1**

© Carnegie Mellon Robotics Academy / For use with VEX Robotics Systems

SWERVBOT BUILDING INSTRUCTIONS

SWERVBOT BUILDING INSTRUCTIONS

USING THE VEX CORTEX

© Carnegie Mellon Robotics Academy / For use with VEX Robotics Systems | Swervebot 1.1 • 1

Reference

Thresholds with Natural Language

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 sensor value (usually a sensor value) falls in relation to the threshold.

Threshold = 25 cm

Green (above threshold) | **Red** (below threshold)

If you look at the image, it shows an LED using an Ultrasonic Ranging Sensor. The threshold in this case is 25 centimeters. We can create behaviors that tell the robot to go forward until the Ultrasonic Ranging Sensor is counting greater than 25 centimeters.

```

1. forward(1);
2. wait(1000);
   
```

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

Calculated Thresholds

Some sensors, like the Ultrasonic Ranging Sensor and Potentiometer, provide the same set of values no matter what environment the robot is in. For the most part, their thresholds can simply be chosen based on their application. Other sensors, like the Light Sensor and Line Tracking Sensor, will provide very different values based on the environment they're in, due to factors such as the amount of ambient light or the type of surface the robot is tracking. Sensors that provide a different range of values based on their environment need to have their thresholds calculated.

For example, 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 Line Follower Sensor value of the light surface. (For more information on finding sensor values, reference the ROBOTC Debugger document.)
2. Measure the Line Follower 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: $\text{light value} + \text{dark value} = \text{threshold}$

© Carnegie Mellon Robotics Academy / For use with VEX Robotics Systems | Thresholds

Sensing/Line Tracking (continued)

Physical Robot Object Programming Challenges

Sensing

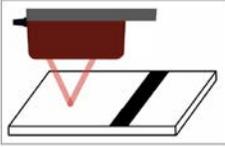
Forward until Dark

In this lesson, you will use a VEX Line Tracking Sensor and the Threshold you calibrated to cause the robot to move forward until it detects a dark surface, and then stop.

Let's review: there is a set of three VEX Line Tracking Sensors. Each contains an infrared LED and an infrared light sensor.



The Line Tracking Sensors can detect the basic colors of objects and surfaces by sensing directly at them at close range. They do so by illuminating a surface with its infrared LED and then measuring how much light is reflected back.



© Carnegie Mellon Robotics Academy / For use with VEX Robotics System

Programming Challenge

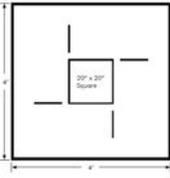
Robo 500

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
- Sensors (or cutting tool)
- Rule (or straight edge)

Board Specifications



Note: Diagrams are not drawn to scale.

© Carnegie Mellon Robotics Academy / For use with VEX Robotics System

Programming Challenge

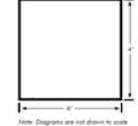
Table Bot

Challenge Description
Program your robot to move forward until it detects the edge of a surface, using the light sensor. The robot should then stop, back up, turn, and square moving forward until it detects another edge, repeating this behavior indefinitely in order to accomplish this task, the robot may need to be placed on a low-friction level.

Materials Needed

- A surface no higher than a few inches off of the ground

Board Specifications



Note: Diagrams are not drawn to scale.

© Carnegie Mellon Robotics Academy / For use with VEX Robotics System

Programming Challenge

Robo-Slalom Challenge Level 2

Challenge Description
To complete this challenge, you will need to program your robot to travel from the Start zone to the Finish zone by following the curved line. Your robot must stop on its own at the Finish zone, and may not bump any of the obstacles along the curved line. You may use feedback from your encoders or timers to control how long the robot follows the line.

Materials Needed

- A flat, open area for the robot to traverse
- Black electrical tape to create the curved line
- Several objects (bottle caps work)

Course Layout



© Carnegie Mellon Robotics Academy / For use with VEX Robotics System

Programming Challenge

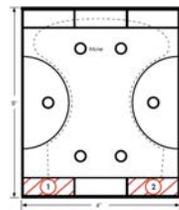
Minifield Traversal

Challenge Description
To ensure the safety of your robot, make sure it doesn't touch any wires (Syntronium copper is a bad idea) or the board itself. 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
- Sensors (or cutting tool)
- Syntronium tape
- Rule (or straight edge)

Board Specifications



Note: Diagrams are not drawn to scale.

- Robot must begin from this start location.
- Robot must finish here.

© Carnegie Mellon Robotics Academy / For use with VEX Robotics System

Programming Challenge

Robo-Slalom Challenge Level 3

Challenge Description
To complete this challenge, you will need to program your robot to travel from the Start zone to the Finish zone by following the curved line. Your robot must stop on its own at the Finish zone, and may not bump any of the obstacles along the curved line. Consider breaking the curved line into several different line following behaviors according to how curved the different sections of the line are.

Materials Needed

- A flat, open area for the robot to traverse
- Black electrical tape to create the curved line
- Several objects (bottle caps work)

Course Layout



© Carnegie Mellon Robotics Academy / For use with VEX Robotics System

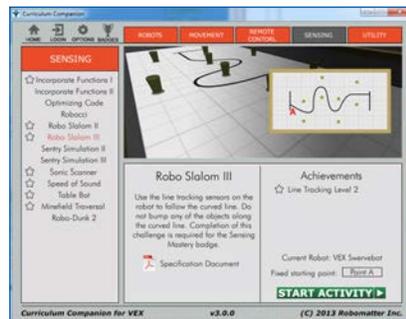
Robot Virtual World Programming Challenges



Curriculum Companion for VEX v3.0.0 (C) 2013 Robomatter Inc.



Curriculum Companion for VEX v3.0.0 (C) 2013 Robomatter Inc.



Curriculum Companion for VEX v3.0.0 (C) 2013 Robomatter Inc.

Sensing/Turn for Angle



Robots separate themselves from simpler machines through their ability to detect and respond to their surroundings. In this section, you will learn how to take advantage of the multiple sensors available to the VEX Cortex.

[EXPAND ALL](#)

	1. Challenge	▼
	2. Limiting the Arm	▼
	3. Behaviors and Functions	▼
	4. Forward until Near	▼
	5. Line Tracking	▼
	6. Turn for Angle	▲
	Using the Gyro Sensor to make more accurate turns.	
	1. The Gyro Sensor	
	2. Turn for Angle 1	
	3. Turn for Angle 2	
	7. Using the LCD	▼

This lesson set will teach students how to program the VEX Gyro Sensor

The Gyro Sensor Video - This video introduces the student to how the Gyro Sensor works.

Sensing Turn for Angle Part 1 - Part one of a video set that teaches how to reset and program the gyro sensor to work with your VEX robot.

Sensing Turn for Angle Part 2 - Part two of a video set that teaches how to reset and program the gyro sensor.

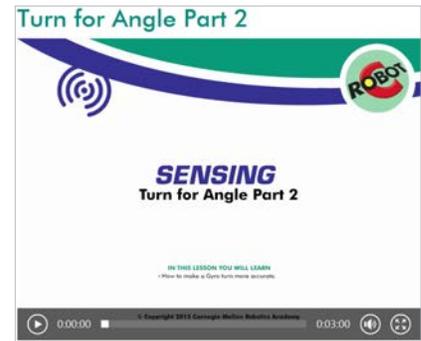
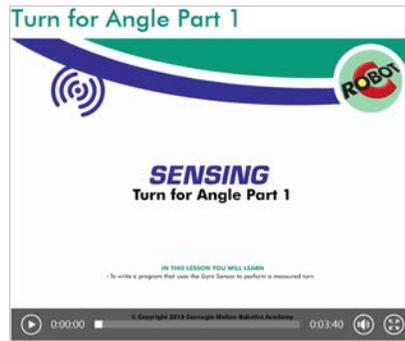
The ROBOT 500 Challenge PDF and RVW - This challenge comes in both a physical and virtual version.

The Minefield Challenge PDF and RVW - This challenge comes in both a physical and virtual version.

Additional resources for the Sensing Turn for Angle lesson set can be found on the next page.

Sensing/Turn for Angle (continued)

Instructional Videos



Physical Robot Programming Challenges

Programming Challenge

Robo 500

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
- Ruler (or straight edge)
- Scissors (or cutting tool)

Board Specifications

Note: Diagrams are not drawn to scale.

© Carnegie Mellon Robotics Academy / For use with VEX Robotics Systems

Programming Challenge

Minefield Traversal

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

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 VEX Robotics Systems

Virtual Robot Programming Challenges

Current Robot: Clawbot IQ
Fixed starting point: **Point A**
START ACTIVITY

Current Robot: Clawbot IQ
Fixed starting point: **Point A**
START ACTIVITY

Sensing/Using the LCD

SENSING

Robots separate themselves from simpler machines through their ability to detect and respond to their surroundings. In this section, you will learn how to take advantage of the multiple sensors available to the VEX Cortex.

COLLAPSE ALL

- 1. Challenge ▲
- 2. Limiting the Arm ▲
- 3. Behaviors and Functions ▲
- 4. Forward until Near ▲
- 5. Line Tracking ▲
- 6. Turn for Angle ▲
- 7. Using the LCD ▼
 - In this chapter you will learn how to use the VEX LCD.
 - 1. Intro to the LCD
 - 2. Displaying a Message
 - 3. Continually Updating the Display

This lesson set will teach students how to program the VEX LED Screen

The VEX LCD Video - This video introduces the many features of the LCD screen.

Displaying Sensor Values on the VEX LCD Video - This video teaches the student how to display continually changing values to the LCD.

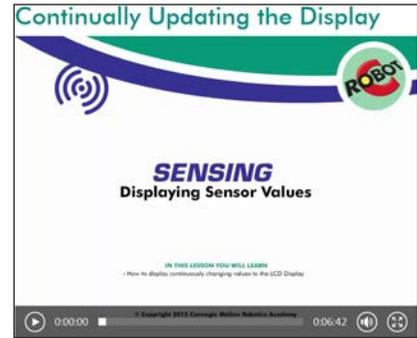
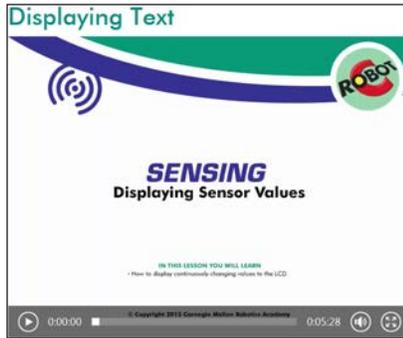
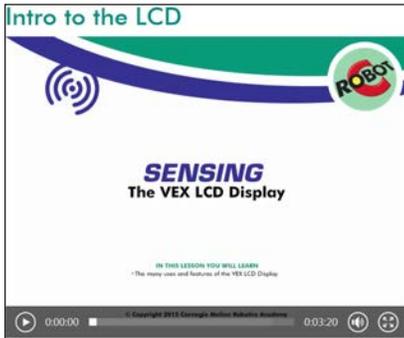
Continually Updating the VEX LCD Video - This video teaches the student how to display continually changing values to the LCD.

The LCD Display Handout PDF - This handout was produced by Innovation First and explains how to connect the LCD to the VEX controller

Additional resources for the Using the LCD lesson set can be found on the next page.

Sensing/The LCD Display (continued)

Instructional Videos



LCD Handout

Logic Accessories

LCD Display

Use the VEX LCD Display Module to receive real-time feedback from your robot to perform live debugging. View multiple stored program configurations and select between them or provide additional user-input to your robot.

INSERT THIS PAGE at the back of the Logic Chassis or your VEX Inventor's Guide.

YOU MUST HAVE A PROGRAMMING KEY TO USE THIS MODULE!

There are two interconnect ports on the side of the LCD module. The two interconnect ports are labeled RX and TX. The RX and TX ports are labeled with respect to the device that connects to the LCD Display Module.

Example: Data is flowing out of the RX port on the LCD Display Module; therefore, the RX (output) port on the LCD Display Module should connect to the RX (input) port on the VEX PIC Microcontroller. Likewise, the TX (input) port on the LCD Display Module should connect to the TX port (output) on the VEX PIC Microcontroller.

Use standard VEX Keyed PWM Cables for connecting the LCD Display Module to the VEX PIC Microcontroller. Use the VEX Serial Y-Cable for connecting the LCD Display Module to the VEX Cortex Microcontroller. The Y section of the Serial Y-Cable with the Yellow wire connects to the RX port and the Y section with the White wire connects to the TX port on the LCD Display Module.

There are three pushbutton switches located on top of the VEX LCD Display Module. These three pushbutton switches can be configured by the user through software.

For help on programming or connecting the LCD Display Module, please refer to your programming software.

Limited 90-day Warranty
This product is warranted by Innovation First against manufacturing defects in material and workmanship under normal use for exactly 90 days from the date of purchase from authorized Innovation First dealers. For complete warranty details and provisions, check with your dealer.

Innovation First, Inc.
1513 IH 30 W
Coppell, TX 75012

For More Information, and additional Parts & Pieces, refer to www.VEXrobotics.com

9219

INVENTOR'S GUIDE INSERT LCD Display • 1

Engineering

HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING ENGINEERING REFERENCE

ENGINEERING

No matter what career you choose, safety, managing projects, and applying engineering processes to solve problems will be important. This section provides helpful information on each and more.

[EXPAND ALL](#)

- 1. Safety
- 2. VEX Hardware
- 3. Engineering Process
- 4. Competition Programming
- 5. Rubrics

Engineering

The Engineering section is broken into 5 sections and includes lots of handouts, instructional videos, and rubrics for assessment.

Engineering/Safety



No matter what career you choose, safety, managing projects, and applying engineering processes to solve problems will be important. This section provides helpful information on each and more.

[EXPAND ALL](#)

	1. Safety	<p>Safety is an "Attitude" that is valued across all types of industry. This section of the curriculum provides resources and activities designed to instill a safe attitude in all students working with robotics.</p> <ul style="list-style-type: none"> 1. Safety Documents 2. Safety Tests
	2. VEX Hardware	
	3. Engineering Process	
	4. Competition Programming	
	5. Rubrics	

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.

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.

Safety continued next page.



Engineering/Safety continued

Handouts

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. All 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. Do not 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 of 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 to believe that personal safety for themselves and their classmates is 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 appropriate to content.
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 resources are found in written teacher guides.
7. Administrators and parents should be notified if any student behaves in a manner that puts themselves or other students at risk.

Laboratory/Physical Setting

Safe teaching and learning conditions begin with organized classroom facilities. This means that proper attention must be paid to adequate space, appropriate ventilation, properly installed electrical outlets, sinks, lighting, ventilation and storage.

Fire extinguishers, workstations and mobile lab carts should be appropriately sized and adequately spaced. Fully line to 60 square feet (2.5-3 square meters) of working space per student should be provided. Class sizes should not exceed 24 students.

VEX® Cortex® Video Trainer using ROBOTC®

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 safety 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. Aisleward 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 the shipping instructions and receipt 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.

VEX® Cortex® Video Trainer using ROBOTC®

Safety

Electrical Safety

Safety education must be an integral part of technology/engineering education. In order to prevent injury to students 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 meet 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 control the adverse condition.

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

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 it may lead to produce severe burns and tissue damage. The physical condition of the victim is another factor which has 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. Being careless or careless about equipment led to the 46, while failure to de-energize equipment, using tools or equipment on 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, cables, conductors, plugs and fuses, and electric extension cords. A variety of unsafe conditions involving the different types of equipment may electrical hazards. Some of the common defects of tools and equipment are listed as follows:
 - a. Improperly grounded equipment (ground was missing, broken or improperly connected);
 - b. Open conductors, switch boxes, damaged or worn connections, and exposed live wires;
 - c. Insulation which is defective, missing, worn, frayed, wet, oily or deteriorated, creating short circuit possibilities and emerging equipment frames;
 - d. Defective switches, receptacles, extension cords, and lamp sockets;
 - e. Dirty motor windings, improperly adjusted switches, and worn connections;
 - f. Improperly connected power tools and defective insulation in portable tools;
 - g. Broken housings, loose or shifting machine parts which might contact and energize hot or machine frames and expose "live" surfaces to operators.

VEX® Cortex® Video Trainer using ROBOTC®

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 meet 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 with a position that there is no danger of their slipping into the cutter or moving parts.
12. Keep machine clear of tools, blocks and other items.
13. Keep the floor around machines clear of liquids, scraps, tools and material.
14. Cover the machine your unattended attention while you are using it. Do not look away to talk to others.
15. Never lean or hang on any machine.
16. Do not use extension cords for permanent connections.
17. Notify your instructor of any breakdown or malfunction.
18. Allow all machines to come to a complete stop before removing work or making a new setup.

VEX® Cortex® Video Trainer using ROBOTC®

Safety

Safety Attitude Test

What's your "OH" safety score? What is your attitude in these situations?

1. If you are upset or angry, do you "vent" to "OH" before you take action?
 Yes No
2. Do you use the safe way?
 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 report safety problems?
 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 "last" safety habits are?
 Yes No

VEX® Cortex® Video Trainer using ROBOTC®

Safety

General Safety Test

1. Freedom from danger or hazards is defined as _____.
2. The biggest hazards in most accident cases are poor safety _____.
3. List five attitudes that cause accidents: _____
4. _____
5. _____
6. _____
7. _____
8. Safety is your _____ responsibility.
9. Should situations might produce hazards and accidents in the office environment?
 True False
10. Safety is a(n) _____ of mind.
11. Accidents are a result of _____ and _____.
12. _____ which you can eliminate.
13. Accidents are generally caused by: _____.
14. Do your own safety report by knowing how to _____ accidents.
15. Self-control, sound judgment, and the desire to be safe are examples of _____ attitudes.

VEX® Cortex® Video Trainer using ROBOTC®

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.

Safety

ROBOTICS LAB Student Safety Inspection Sheet

Break your one team and complete the safety inspection sheet to the left. 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 appropriate 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 non-slip 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-glass lighting is provided for all work areas according to state regulations.
12. S U NA Stairways within working laboratories have safe tread and rise with unobstructed aisles and with approved railings.
13. S U NA Runglins and ladders are color coded.
14. S U NA Two safety regulated marked exits are available from each major laboratory.
15. S U NA Facilities are tight, clean, clear and conducive to good instruction.
16. S U NA Machine operation regulations and safety procedures are posted conspicuously in all major shop areas.
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.

VEX® Cortex® Video Trainer using ROBOTC®

Safety

Safety QUIZ

Put a check - 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 have things where they are.
 True False
3. You should take off jewelry when using power tools.
 True False
4. You should be 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 make 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

VEX® Cortex® Video Trainer using ROBOTC®

Engineering/VEX Hardware

HOME
FUNDAMENTALS
SETUP
MOVEMENT
REMOTE CONTROL
SENSING
ENGINEERING
REFERENCE

ENGINEERING

No matter what career you choose, safety, managing projects, and applying engineering processes to solve problems will be important. This section provides helpful information on each and more.

+ EXPAND ALL

1. Safety
▼

2. VEX Hardware
▲

The VEX Cortex Robotics System consists of a microcontroller, a set of sensors and motors, and lots of parts. The documents in this chapter contain valuable information about how the system works.

- ▶ 1. Motors and Building Materials
- ▶ 2. Sensors and Displays

3. Engineering Process
▼

4. Competition Programming
▼

5. Rubrics
▼

VEX Hardware

The VEX Cortex Robotics System consists of a micro controller, a set of sensors and motors, and lots of parts. The documents on the next couple of pages contains valuable information about how the system works. These documents are available online for free student access and can be used as reference documents or assigned as homework study guides.

Structure

Structure

Table of Contents:

Introduction to Structure Subsystem	2.2
Concepts to Understand	2.7
Subsystem Interactions	2.17

VEX Inventor's Guide 2 • 1

Motion

Motion

Table of Contents:

Introduction to the Motion Subsystem	3.2
Concepts to Understand	3.8
Subsystem Interactions	3.26

VEX Inventor's Guide 3 • 1

Sensor

Sensor

Table of Contents:

Introduction to the Sensor Subsystem	5.2
Concepts to Understand	5.3
Subsystem Interactions	5.11

VEX Inventor's Guide 5 • 1

Fundamentals/VEX Hardware (continued)

http://www.education.rec.ri.cmu.edu/products/teaching_robotc_cortex/index.html

RECHARGEABLE BATTERY VOLTAGE

2-WIRE MOTOR

DETAIL A SCALE 1:1

DATA CLOCK, **DATA IN**, **DATA OUT**, **GROUND**, **5V**, **3.3V**, **5V**, **GROUND**

DETAIL B SCALE 1:1

USB-A INTERFACE

7.2V MAIN BATTERY, **ON/OFF SWITCH**, **7.2V MAIN BATTERY**

DETAIL C SCALE 1:1

SPEAKER OUTPUT, **DIGITAL INPUTS/OUTPUTS**, **ANALOG INPUTS**

Manufacturer: VEX Cortex Pinout
 Part #: 274214 Rev1 Pinout
 Rev: BDR
 Date: 12/1/2010
 ALL DIMENSIONS ARE IN INCHES. www.VEXROBOTICS.COM

Logic Accessories

LCD Display

Use the VEX LCD Display Module to receive real-time feedback from your robot to perform fine-tuning. View multiple stored program configurations and select between them or provide additional user-input to your robot.

INSERT THIS PAGE at the back of the Logic Chapter in your VEX Inventor's Guide.

YOU MUST HAVE A PROGRAMMING KIT TO USE THIS MODULE!

There are two interconnect ports on the side of the LCD module. The two interconnect ports are labeled RX and TX. The RX and TX ports are labeled with respect to the device that connects to the LCD Display Module.

Example: Data is flowing out of the RX port on the LCD Display Module; therefore, the RX (output) port on the LCD Display Module should connect to the RX (input) port on the VEX PIC Microcontroller. Likewise, the TX (input) port on the LCD Display Module should connect to the TX port (output) on the VEX PIC Microcontroller.

Use standard VEX Keyed PWM Cables for connecting the LCD Display Module to the VEX PIC Microcontroller. Use the VEX Serial Y-Cable for connecting the LCD Display Module to the VEX Cortex Microcontroller. The Y section of the Serial Y-Cable with the yellow wire connects to the RX port and the Y section with the white wire connects to the TX port on the LCD Module.

There are three pushbutton switches located on top of the VEX LCD Display Module. These three pushbutton switches can be configured by the user through software.

For help on programming or connecting the LCD Display Module, please refer to your programming software.

Manufacturer: VEX
 Part #: 274214 Rev1 LCD Display
 Rev: BDR
 Date: 12/1/2010
 ALL DIMENSIONS ARE IN INCHES. www.VEXROBOTICS.COM

Motion Accessories

Motor Kit

Continuous Rotation Motor Kit

Additional motors can be used to provide a robot with additional functionality; more motors means more features. For more information on the VEX Continuous Rotation Motors and how to use them within the VEX Robotics Design System, refer to the Motion section of the VEX Inventor's Guide.

INSERT THIS PAGE at the back of the Motion Chapter in your VEX Inventor's Guide.

Motor Specs:

Free Speed	100 RPM @ 7.0 Volts
Stall Torque	4.5 in-oz (approximate)
Stall Current	3.6 A (approximate)
Free Current	190 mA (approximate)

Notes:

- Each VEX Motor contains a thermal limiter which will "trip" when motor loading exceeds 3 Amps.
- A cool motor that is loaded to a stall condition will trip in less than 30 seconds.
- Once tripped the thermal limiter will cut power to the motor until loading is completely removed and the motor is allowed to cool for at least a few seconds.
- Since this is a thermal limit, it will be easier to trip when it is hot. After it trips once, it will be easier to trip a second time.
- Allow the motor to fully cool to re-engage the limit between trips.

Motor with Club x 1

Club 6-32 x 1/2" x 2

Club Nut x 1

Replacement Gears x 2

Manufacturer: VEX
 Part #: 274214 Rev1 Motor Kit
 Rev: BDR
 Date: 12/1/2010
 ALL DIMENSIONS ARE IN INCHES. www.VEXROBOTICS.COM

motion accessories

servomotor kit

Servomotor

As explained in the Motion Subsystem section of the Inventor's Guide, servomotors are a type of motor that can be directed to turn to face a specific direction, rather than just spin forward or backward.

INSERT THIS PAGE at the back of the Motion Chapter in your VEX Inventor's Guide.

servomotor x 1

screw 6-32 x 1/2" x 2

screw 6-32 x 1/4" x 2

club nut x 1

replacement gears x 2

Manufacturer: VEX
 Part #: 274214 Rev1 Servomotor Kit
 Rev: BDR
 Date: 12/1/2010
 ALL DIMENSIONS ARE IN INCHES. www.VEXROBOTICS.COM

Motion Accessories

Motor Controller 29

Motor Controller Module

The Motor Controller 29 replicates the speed of a VEX motor based on a signal it receives from a VEX Microcontroller. This allows for control of any VEX motor that doesn't have a built-in motor controller, such as the VEX 2-wire motor.

INSERT THIS PAGE at the back of the Motion Chapter in your VEX Inventor's Guide.

Motor Controller Specs:

Max Current	3 amps @ 8.5v (full forward or full reverse)
Idle Current	12 mA @ 8.5v (typical)

Notes:

- Each VEX Motor Controller Module can drive one 2-wire motor.
- Max motor stall current: 3 amps.
- Max battery voltage: 8.5v (with fully charged 7.2v battery).
- Connect 2-wire socket to motor 2-wire plug.
- Easy reverse, range connection to motor plug.

How To Connect

- Connect Motor Controller 2-wire plug to microcontroller motor port.
- Connect 2-wire motor plug to motor controller 2-wire jack.
- To reverse, range connection.

Manufacturer: VEX
 Part #: 274214 Rev1 Motor Controller 29
 Rev: BDR
 Date: 12/1/2010
 ALL DIMENSIONS ARE IN INCHES. www.VEXROBOTICS.COM

Motion Accessories

2-Wire Motor 269

The 2-Wire Motor 269 replaces the 3-Wire Motor as the standard VEX motor. All of the internal gears are made from a steel alloy, which means the clubnut and replacement gears are no longer required. The 2-wire motor can be directly connected to the Cortex and ARM9 microcontroller internal motor controllers. An external motor control module is required to connect the 2-wire motor to the PIC Microcontroller V.S. External motor control modules can also be used with the Cortex and ARM9 microcontrollers.

INSERT THIS PAGE at the back of the Motion Chapter in your VEX Inventor's Guide.

Motor Coupler

The 2-Wire Motor 269 kit includes the new shaft coupler which can be used in place of the clubnut to connect the motor to VEX shafts. The coupler can also be used to connect VEX shafts together.

Motor Specifications

All motor specifications are at 7.2 volts. Actual motor specifications are within 20% of the values below.

Description	Specification
Stall Torque	8.5 in-oz (0.97 B-in)
Free Speed	100 RPM
Stall Current	3.6 Amps
Free Current	0.19 Amps

2 Wire Motor x 1

Motor Coupler x 1

Motor Pin x 1

Manufacturer: VEX
 Part #: 274214 Rev1 Motor 269
 Rev: BDR
 Date: 12/1/2010
 ALL DIMENSIONS ARE IN INCHES. www.VEXROBOTICS.COM

Motion Accessories

2 Wire Motor 393

The 2 Wire Motor 393 provides up to 60% more torque than the standard motor, which still allow more powerful mechanisms and drive bases. All of the internal gears are made from a steel alloy, which means that clubnut and replacement gears are no longer required. The 2-wire motor can be directly connected to the Cortex and ARM9 microcontroller internal motor controllers. An external motor control module is required to connect the 2-wire motor to the PIC Microcontroller V.S. External motor control modules can also be used with the Cortex and ARM9 microcontrollers.

INSERT THIS PAGE at the back of the Motion Chapter in your VEX Inventor's Guide.

High Speed Option

Wound to go faster than the standard motor but still have the same output torque as the standard motor. No problem! The 2 Wire Motor 393 kit can be configured into a "high speed" version. Simply follow the "Gear Change Procedure" step-by-step instructions to increase the output speed by 60%.

Motor Coupler

The 2 Wire Motor 393 kit includes the new shaft coupler which can be used in place of the clubnut to connect the motor to VEX shafts. The coupler can also be used to connect VEX shafts together.

Motor Specifications

All motor specifications are at 7.2 volts. Actual motor specifications are within 20% of the values below.

Description	As Shipped	High Speed Option
Stall Torque	13.5 in-oz (1.48 N-m)	8.4 in-oz (1.05 N-m)
Free Speed	100 RPM	160 RPM
Stall Current	3.6 Amps	3.6 Amps
Free Current	0.19 Amps	0.19 Amps

2 Wire Motor x 1

Motor Coupler x 1

Motor Pin x 1

Manufacturer: VEX
 Part #: 274214 Rev1 Motor 393
 Rev: BDR
 Date: 12/1/2010
 ALL DIMENSIONS ARE IN INCHES. www.VEXROBOTICS.COM

sensor accessories

Optical Shaft Encoder Kit

Basic Optical Shaft Encoders are commonly used for position and motion sensing. Basically, a disc with a pattern of cutouts around the circumference is positioned between an LED and a light detector in the disc rotates, the light from the LED is blocked by a regular pattern. This pattern is processed to determine how far the disc has rotated. If the disc is then attached to a wheel on a robot, it is possible to determine the distance that wheel traveled, based on the circumference of the wheel and the number of revolutions it made.

With the Quadrature Encoder, there are 2 output channels. Only one output can be used as a basic Optical Shaft Encoder. The same quadrature refers to the situation where there are two output channels that are 90 degrees out of phase with each other, being captured by the units. The two output channels of the Quadrature Encoder can be used to indicate both position and direction of rotation.

INSERT THIS PAGE at the back of the Sensor Chapter in your VEX Inventor's Guide.

YOU MUST HAVE A PROGRAMMING KIT TO USE THIS SENSOR!

Optical Shaft Encoder x 2

Screw x 8-32 3/8"

Open Nut x 4

Manufacturer: VEX
 Part #: 274214 Rev1 Optical Shaft Encoder Kit
 Rev: BDR
 Date: 12/1/2010
 ALL DIMENSIONS ARE IN INCHES. www.VEXROBOTICS.COM

Engineering/VEX Hardware (continued)

Handouts

sensor accessories

Potentiometer Kit

The Vex Potentiometer will keep things "on the level". Use this sensor to get an analog measurement of angular position. This measurement can help to understand the position of robot arms, or other mechanisms. To effectively utilize this sensor, users are required to use the Vex Programming Kit.

The Potentiometer is designed with a "1/2 inch" in the center. This hole should slide easily over the Vex square shafts. The Potentiometer also includes (2) "arcs" which are 1/2" from the center hole; these arcs are used for mounting the Potentiometer to the robot structure.

The mounting arcs allow for 90-degree of adjustment to the Potentiometer position. Since the Potentiometer has limited travel, it is important to ensure that the shaft that is being measured by the Potentiometer does not travel more than 240-degrees (the Potentiometer can only measure mechanically about 245-degrees ± 5 and can only measure electrically 250-degrees ± 2%). The adjustment arcs allow the Potentiometer's "range of motion" to be repositioned to match the shaft's range of motion. To measure the motion of something which moves more than 250-degrees, try gearing down the shaft's motion to a secondary shaft (this secondary shaft will move less distance and then measure this secondary shaft).

YOU MUST HAVE A PROGRAMMING KIT TO USE THIS SENSOR!

© 2015 VEX Robotics Inc. www.VEXRobotics.com

sensor accessories

ultrasonic sensor kit

ultrasonic sensor kit

Ultrasonic Sensor Kit
"Ultrasonic" refers to very high frequency sound – sound that is higher than the range of human hearing. Sound, or "Sound Navigation And Ranging," is an application of ultrasonic sound that uses propagation of these high-frequency sound waves to navigate and detect obstacles. Sound has a wide variety of applications and a wide variety of users, from submarines avoiding underwater obstacles to hungry bats looking for their dinner!

INSERT THESE PINS at the back of the Sensor Chamber of your VEX Inverter's Guide.

YOU MUST HAVE A PROGRAMMING KIT TO USE THIS SENSOR!

Insert These Pins at the back of the Sensor Chamber of your VEX Inverter's Guide.

YOU MUST HAVE A PROGRAMMING KIT TO USE THIS SENSOR!

© 2015 VEX Robotics Inc. www.VEXRobotics.com

sensor accessories

line follower kit

line follower kit

Line Follower Kit
A line follower consists of an infrared light sensor and an infrared LED. It works by illuminating a surface with infrared light; the sensor then picks up the reflected infrared radiation and, based on its intensity, determines the reflectivity of the surface in question. Light-colored surfaces will reflect more light than dark surfaces, resulting in their appearing brighter to the sensor. This allows the sensor to detect a dark line on a pale surface, or a pale line on a dark surface.

You can use a line follower to help your robot navigate along a marked path, or in any other application involving discerning the boundary between two high-contrast surfaces. A typical application uses three line follower sensors, such that the middle sensor is over the line your robot is following.

INSERT THESE PINS at the back of the Sensor Chamber of your VEX Inverter's Guide.

YOU MUST HAVE A PROGRAMMING KIT TO USE THIS SENSOR!

© 2015 VEX Robotics Inc. www.VEXRobotics.com

Engineering/Engineering Process

HOME
FUNDAMENTALS
SETUP
MOVEMENT
REMOTE CONTROL
SENSING
ENGINEERING
REFERENCE

ENGINEERING

No matter what career you choose, safety, managing projects, and applying engineering processes to solve problems will be important. This section provides helpful information on each and more.

+ EXPAND ALL

1. Safety
▼

2. VEX Hardware
▼

3. Engineering Process
▲

4. Competition Programming
▼

5. Rubrics
▼

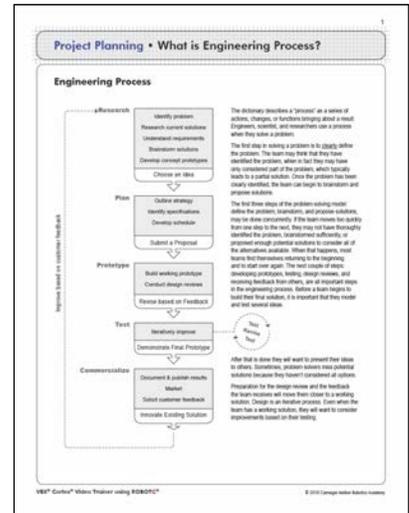
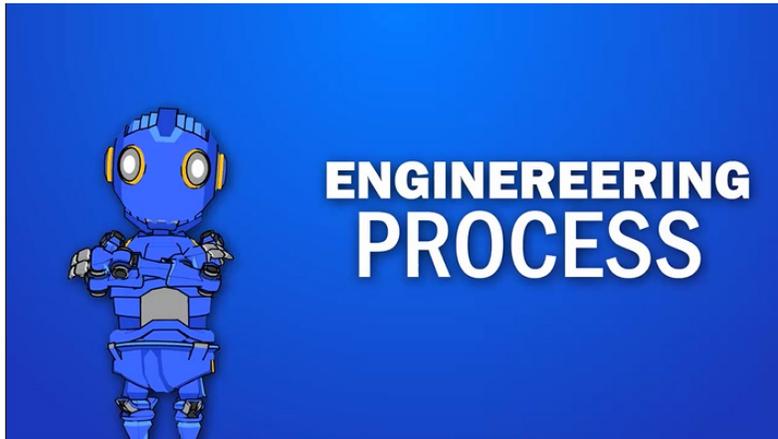
The resources below will help you solve design problems.

- ▶ [1. The Engineering Process](#)
- ▶ [2. Project Planning](#)
- ▶ [3. Engineering Project](#)

Many schools compete in robotic competitions; other schools are using the VEX Cortex and IQ systems to teach engineering. The resources in the engineering section provide students with materials that teach how to manage and solve engineering design problems.

Engineering/Engineering Process *continued*

The Engineering Process



Engineering Process Resources

Engineering Process Video - 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.

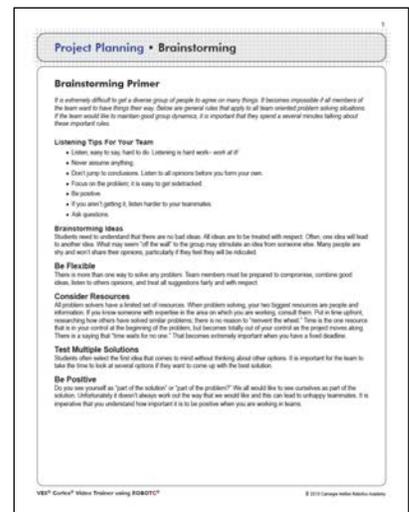
Engineering Process Reference - This three page set of handouts describes steps that engineers use to solve problems, provides a set of definitions for the word "engineering", and describes the iterative nature of design.

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.

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.

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.



Engineering process resources continued next page.

Engineering/Engineering Process *continued*

Handouts

Project Planning • Team Building

First Team Meeting
Your first team meeting sets the tone for your entire project. Here are some guidelines to help it be the great start you're looking for.
Meet people properly. It all starts with the introduction. Be positive, show that you are genuinely interested in meeting your new teammates. You don't get a second chance to make a first impression.

Find things you have in common
You can always find things in common: you go to the same school, have the same classes, and maybe are from the same neighborhood. In the next several paragraphs are things like sports, movies, songs, etc. all the ways that you have the weather. This is why often take jobs at their professional sports teams. They are socially gathering forces that cut across boundaries of race and wealth.

Make meeting conditions good
Make sure you have a large surface to write on, make sure the room is quiet and comfortable, and that there aren't a lot of other distractions.

Everyone can contribute
This can be difficult. Even if you don't agree and think what your team member is saying is foolish, be patient. Cutting someone off is rude, and it's certainly not worth whatever small idea you might make. Don't break someone else's sentences for him or her; they can do it for themselves. And remember: talking louder or faster doesn't make your idea any better.

Please alternatives as questions
Instead of "I think we should do A, not B," try "What if we did A, instead of B?" That allows people to offer comments, rather than defend one choice.

Write things down
Have someone take notes that document the meeting. Be sure to include everyone's ideas.

Find good things to say about other people's ideas
You need not agree with ideas that are presented, but it is important to not shoot someone else's idea down without finding something positive about the idea. Find the "value being" in their idea and then present your concerns. Do not embarrass your teammates.

Be honest but not harsh
Talk with your group members if there is a problem, and talk with your teacher if you think you need help. One of the points of the course is to learn to work together. Disagreements often provide valuable insights, but don't be afraid to raise the issues when they come up.

Avoid conflict at all costs
Other ideas, concepts, and strategies often take a short break. Clear your heads, apologize, and take another stab at it. Apologize for venting your frustration, even if you think someone else was primarily at fault; the goal is to work together, not start a fight. Unlike some other team dynamics were worse. It also has to be an agreement, so be the peacemaker.

VEX® Cortex® Video Trainer using ROBOTC® © 2015 Carnegie Mellon Robotics Academy

Project Planning • Understanding the Problem

Putting it in Context
Solving a problem is much easier when you understand why it is a problem, and not just a fact of life. Before you even start brainstorming solutions, you need to find out a little bit about the issue at hand. It's possible that some of the research you do now will help you make more informed choices later on in the project.

Answer the questions below relating to your project.
What will you be designing or redesigning? Describe it in at least ten full sentences.

Who is the audience for this project? Who wants you to make this?

Will this project be beneficial to anyone outside of the group that it is specifically designed for?

Has this project ever been undertaken before? If not, why not?

Is there a commercial market for the outcome of this project? Can it be manufactured at a reasonable cost?

Will this project have any negative environmental impacts?

Will the product be easy to use? How can we make it so?

VEX® Cortex® Video Trainer using ROBOTC® © 2015 Carnegie Mellon Robotics Academy

Project Planning • Design Reviews

The Design Review
In industry, most projects require teams of people working together to complete the job. Communication is the key to ensuring a successful end to a project. One tool that is used to ensure successful communication within the team is called a design review. During the review, the team is able to talk about technical problems, resources needed, scheduling conflicts, that dates, etc. We will introduce three types of reviews: weekly preliminary and detail design reviews. The goal of the design review is to get feedback from all members of the team and ensure a successful and timely completion of the project.

Weekly Reviews
Teams may need to meet daily in the early stages of a project, but once the work starts, the group will want to meet at least once a week to check on the progress of the project. These meetings give all members of the team a chance to see how the overall project is moving along. Additionally, they allow the project manager to look at the team's resources and workload from an overall perspective. Documentation of progress should be kept and weekly goals should be set for individuals and for the team. The project manager and team leaders should monitor these goals. Assessment of these goals takes place at the weekly design reviews.

Preliminary and Detail Design Reviews
There are two types of reviews: preliminary and detail design reviews. The Preliminary Design Review allows the team to get initial feedback on the overall concept they will use to solve the problem. Interested reviewers are identified from the community. They are invited, the team presents their concepts, and reviewers give feedback and offer suggestions. Depending on the complexity of the problem, it may be appropriate to give reviewers documents to review before they come to the preliminary design review so they are able to properly prepare. In the preliminary design review, reviewers may have concept maps, plans, prototypes, and models that demonstrate possible solutions.

The Detail Design Review integrates feedback from the preliminary design review. If the group is working on a very large project, there may be multiple detail design reviews. This allows for reviewers and other interested parties to get feedback on the progress of the project in regularly scheduled intervals.

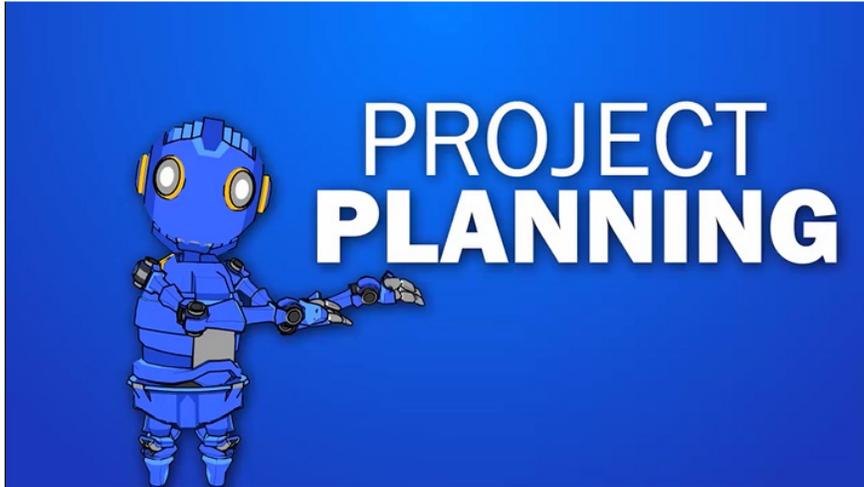
Preparation for the Detail Design Review
Detail design reviews are individual projects. They require proper preparation.

IT IS IMPORTANT THAT DESIGN TEAMS ARE READY FOR REVIEWS.
It is embarrassing to all concerned if team members haven't prepared properly. In the marketplace, detail design reviews are those when stakeholders make decisions about funding projects. The team should make a concerted effort to be properly prepared when they meet with reviewers. If appropriate, the team will want to include examples of the following documentation: a set of detail drawings, assembly drawings, a Gantt chart, metrics and projections, pictures of similar examples, and sample programming logic or control examples (screens & logs).

VEX® Cortex® Video Trainer using ROBOTC® © 2015 Carnegie Mellon Robotics Academy

Engineering/Project Planning

Project Planning

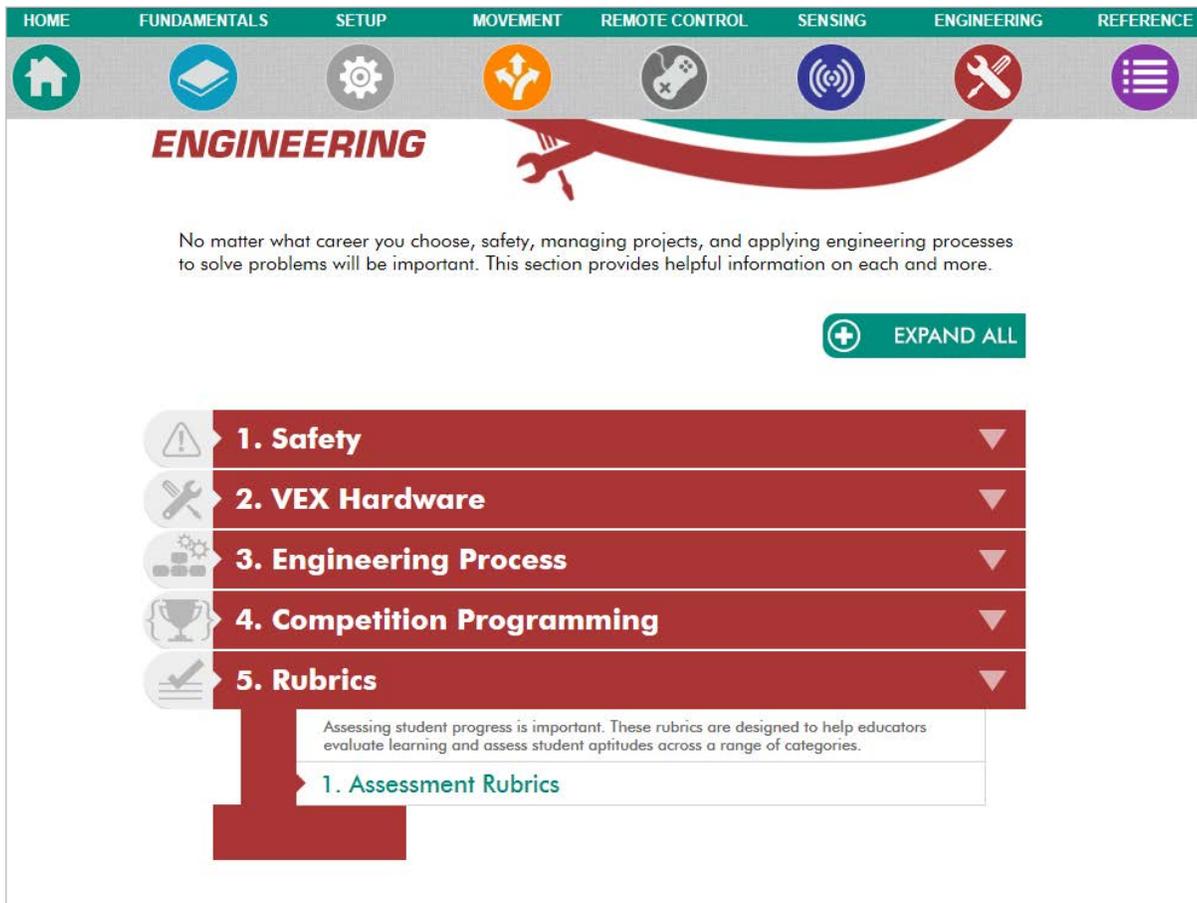


-  Team Building
-  Gantt Chart
-  PERT Chart
-  Organizational Matrix Ideas
-  Recording Progress
-  Preparation for a Competition
-  Planning Your Time

Proper planning is very important to solving any design problem - This section provides many resources designed to help students learn to manage their time.

Project Planning resources can be found on the next page.

Engineering/Assessment Rubrics



HOME FUNDAMENTALS SETUP MOVEMENT REMOTE CONTROL SENSING ENGINEERING REFERENCE

ENGINEERING

No matter what career you choose, safety, managing projects, and applying engineering processes to solve problems will be important. This section provides helpful information on each and more.

[EXPAND ALL](#)

1. Safety
2. VEX Hardware
3. Engineering Process
4. Competition Programming
5. Rubrics
 - Assessing student progress is important. These rubrics are designed to help educators evaluate learning and assess student aptitudes across a range of categories.
 - 1. 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, pre-writing, 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 RFP 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.

Engineering/Assessment Rubrics continued

Rubrics for Engineering Journal Assessment

ROBOTC 1

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 whole course 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 her or his own material organized.

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

Each student's Engineering Journal contains:

- Class handouts
- Daily log and notes
- All completed and returned assignments
- Final bound or revised set 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.)
 - Copy logs, one per stop of individual work, etc. - group journal keeper only
 - All major project deliverable completed, 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:
 - not per model for assignments that are kept
 - damaged or sloppy (professionalism)
- When requested, students should hand in journals.
 - This is the preferred method for collecting work on due assignments due. Profilers apply for groups or individuals who are not prepared.
- Journal contents should be graded according to their own criteria.
 - Assignments should be graded according to their own criteria.
 - Student and journal handouts should be kept for convenience.
- Notes and log are a student's evidence of work done on a daily basis.
 - Self and peer-reported student records are how work habits are tracked.
 - Examples:
 - Effective use of time
 - Good planning and preparation.

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy

Rubrics for Internal Design Review

ROBOTC 1

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Timeliness (10%)	<ul style="list-style-type: none"> Design Candidate Sheets are completed and turned in for each design. Designs are completed on time. Designs are completed on time. Group is present and ready to begin on time. 	<ul style="list-style-type: none"> Design Candidate Sheets are completed but not on time. Designs are completed on time. Group is present and ready to begin on time. 	<ul style="list-style-type: none"> Most Design Candidate Sheets are completed. Designs are completed on time. Group is present. 	<ul style="list-style-type: none"> Design Candidate Sheets are not completed. Designs are not completed. Group is not present.
Discussion (30%)	<ul style="list-style-type: none"> Group follows good meeting and teamwork procedures. Discussion remains professional and on time and on subject. Discussion proceeds efficiently. Group is able to discuss relevant aspects of the robot design. 	<ul style="list-style-type: none"> Group follows good meeting and teamwork procedures. Discussion remains professional and on time and on subject. Discussion proceeds efficiently. Group is able to discuss relevant aspects of the robot design. 	<ul style="list-style-type: none"> Group follows few meeting and teamwork procedures. Discussion does not have a professional tone. Group does not proceed efficiently. Group rarely discusses relevant aspects of robot design. 	<ul style="list-style-type: none"> Group does not work as a team. Little discussion occurs. Group does not have a professional tone. Group does not discuss relevant aspects of robot design.

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy

Rubrics for External Design Review

ROBOTC 1

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Timeliness (10%)	<ul style="list-style-type: none"> Prototype is fully functional at the time of review. Group is present and ready to begin on time. 	<ul style="list-style-type: none"> Prototype is semi-functional but not functional at the time of review. Group is present and ready to begin on time. 	<ul style="list-style-type: none"> Prototype is not built or not functional at the time of review. Group is present and ready to begin on time. 	<ul style="list-style-type: none"> Prototype is not built or not functional at the time of review. Group is not present.
Presentation (15%)	<ul style="list-style-type: none"> Discussion remains professional and on time and on subject. Discussion proceeds efficiently. Group is able to discuss relevant aspects of the robot design. 	<ul style="list-style-type: none"> Discussion remains professional and on time and on subject. Discussion proceeds efficiently. Group is able to discuss relevant aspects of the robot design. 	<ul style="list-style-type: none"> Discussion does not have a professional tone. Discussion does not proceed efficiently. Group rarely discusses relevant aspects of robot design. 	<ul style="list-style-type: none"> Discussion occurs during the presentation but is unprofessional. Students show lack of understanding of group design and design of robot. Students may articulate some ideas of the project, but do not focus on major aspects of robot design. Group does not focus discussion.
Project Management (30%)	<ul style="list-style-type: none"> Development is in line with milestones submitted with proposal. Group members have defined responsibilities which were followed to a substantial degree. Development is present and ready to be demonstrated. 	<ul style="list-style-type: none"> Development is in line with milestones submitted with proposal. Group members have defined responsibilities which were followed to a substantial degree. Development is present and ready to be demonstrated. 	<ul style="list-style-type: none"> Group members have defined responsibilities which were followed to a substantial degree. Development is present but not ready to be demonstrated. 	<ul style="list-style-type: none"> Timeline is not followed. Group members have defined responsibilities which were not followed to a substantial degree. Development is not present and ready to be demonstrated.

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy

Rubrics for Presentations

ROBOTC 1

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Use of Multimedia Technology	<ul style="list-style-type: none"> Student demonstrated how to use multimedia technology. The presentation was clear. The graphics were clear. The sequence of the presentation was well thought out. Animation was integrated. Sound was used appropriately. Complete. 	<ul style="list-style-type: none"> Student demonstrated how to use multimedia technology. The presentation was clear. The graphics were clear. The sequence of the presentation was well thought out. Animation was integrated. Sound was used appropriately. Complete. 	<ul style="list-style-type: none"> Multimedia technology use needs work. The use of multimedia technology was a distraction rather than help. Incomplete. 	<ul style="list-style-type: none"> Multimedia technology didn't support topic. Picture were not clear and didn't seem to be part of the program. Incomplete.
Content Analysis	<ul style="list-style-type: none"> Content was clear. Presentation was organized. Project was fully described. Relevant content related to the presentation. Presentation was professional. 	<ul style="list-style-type: none"> Content was good. Presentation was organized. Project was fully described. Relevant content related to the presentation. Presentation was professional. 	<ul style="list-style-type: none"> Content of presentation lacked clarity. Presentation lacked organization and didn't have a unified theme. Presentation didn't use proper terminology. 	<ul style="list-style-type: none"> Content of presentation lacked clarity. Presentation lacked organization and didn't have a unified theme. Presentation didn't use proper terminology.

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy

Rubrics for Project Proposal Assessment

ROBOTC 1

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Timeliness (10%)	<ul style="list-style-type: none"> All required elements are produced on time. 	<ul style="list-style-type: none"> Most required elements are produced on time. 	<ul style="list-style-type: none"> Few required elements are produced on time. 	<ul style="list-style-type: none"> Required elements are not turned in.
Presentation (20%)	<ul style="list-style-type: none"> Proposed in well written with no grammatical or spelling errors. Proposed has been reviewed at least once. All required elements are included. Timeline and charts are written clearly with the units of measure clearly marked on axes. 	<ul style="list-style-type: none"> Proposed in well written with no grammatical or spelling errors. Proposed has been reviewed at least once. All required elements are included. Timeline and charts are written clearly with the units of measure clearly marked on axes. 	<ul style="list-style-type: none"> Proposed in fairly well written with many grammatical or spelling errors. Proposed has been reviewed. Most required elements are included. Timeline and charts are not written clearly with many grammatical or spelling errors and misspellings. 	<ul style="list-style-type: none"> Proposed has many grammatical or spelling errors. Proposed has not been reviewed. Few required elements are included. Timeline and charts are not written clearly with many grammatical or spelling errors and misspellings.
Practicality (30%)	<ul style="list-style-type: none"> Proposed solution demonstrates understanding of real world constraints (i.e. laws of physics, time). Timeline specifies most due dates for required elements. Materials list is reasonable. Proposed clearly identifies the problem to be solved. 	<ul style="list-style-type: none"> Proposed solution demonstrates understanding of real world constraints (i.e. laws of physics, time). Timeline specifies most due dates for required elements. Materials list is reasonable. Proposed clearly identifies the problem to be solved. 	<ul style="list-style-type: none"> Proposed solution demonstrates understanding of real world constraints (i.e. laws of physics, time). Timeline specifies most due dates for required elements. Materials list is reasonable. Proposed clearly identifies the problem to be solved. 	<ul style="list-style-type: none"> The timeline indicated is not reasonable. Materials list is not reasonable. The connection made between the proposed solution and the problem is not clear.

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy

Rubrics for Robotics Explorations Assessment

ROBOTC 1

	(4) Advanced - A	(3) Proficient - B	(2) Basic - C	(1) Below Basic - D or E
Build/Program Test Robot	<ul style="list-style-type: none"> Robot is built accurately with no mistakes. Robot is programmed accurately with few errors. 	<ul style="list-style-type: none"> Robot is built with some mistakes. Robot is programmed with some errors. 	<ul style="list-style-type: none"> Robot is built with some mistakes. Robot is programmed with some errors. 	<ul style="list-style-type: none"> Robot is built with many mistakes. Robot is programmed with many errors.
Data Analytical/Scientific Method	<ul style="list-style-type: none"> Make reasonable predictions based on prior knowledge. Correctly gather and record data. Construct a bar graph illustrating results with few errors. 	<ul style="list-style-type: none"> Make predictions based on prior knowledge. Correctly gather and record data. Construct a bar graph illustrating results with few errors. 	<ul style="list-style-type: none"> Predictions are based on prior knowledge. Correctly gather and record data. Construct a bar graph illustrating results with some errors. 	<ul style="list-style-type: none"> No predictions are made. Data is not recorded correctly. Bar graph is inaccurate.
Writing an Analytical Paragraph	<ul style="list-style-type: none"> 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 summarizing the results of the experiment. 	<ul style="list-style-type: none"> 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 summarizing the results of the experiment. 	<ul style="list-style-type: none"> 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 summarizing the results of the experiment. 	<ul style="list-style-type: none"> 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 summarizing the results of the experiment.

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy

Rubrics for Work Habits Evaluation

ROBOTC 1

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 fully in the cleanup program.						
Work Habits Point Total						

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy

Rubrics for Workplace Competencies Evaluation

ROBOTC 1

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 Honored	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 Problem solver	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 Shows leadership	Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)
14 Practices good grooming and dresses appropriately	Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)
15 Meets deadlines while producing quality	Always (7) - Usually (5) - Sometimes (3) - Seldom (1) - Never (0)
Work Habits Point Total	

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy

Rubrics for Robotics Class Writing

ROBOTC 1

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 have in great detail the following:

- 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.
- An outline of how 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 that is on the back of your brainstorming sheet?
- Final draft of your paper. Double space your paper so that it will be easier to edit. You can add on the same page that you have in - just make sure 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 always? Does the information you write clearly support what you are trying to say? Did you proofread your paper for errors?
- One point for spelling.

Spn A 4pts B 3pts C 2pts D

VEX® Cortex® Video Trainer Using ROBOTC® © 2013 Carnegie Mellon Robotics Academy



Reference Index

Video Lessons

Fundamentals

1. Programmer and Machine
2. Planning and Behaviors
3. ROBOTC Rules Part 1
4. ROBOTC Rules Part 2

Setup

1. Updating the VEX Cortex Firmware(Wireless)
2. Updating the VEXnet Joystick
3. Updating the VEX Cortex Firmware(Wired)
4. Download Sample Program over VEXnet 1
5. Download Sample Program over VEXnet 2
6. Download your First Program
7. Camera Operation in RVW
8. Measurement Toolkit

Movement

1. Labarynth Challenge
2. Program Dissection
3. Reversing Motor Polarity
4. Renaming Motors
5. Timing
6. Motor Power Levels
7. Turn and Reverse
8. Manual Straightening
9. Shaft Encoders
10. Forward for Distance Part 1
11. Forward for Distance Part 2
12. The Sensor Debug Window
13. Forward and Turning
14. Automated Straightening Part 1
15. Automated Straightening Part 2
16. Values and Variables Part 1
17. Values and Variables Part 2
18. Forward for Distance IME
19. Principals of PID
20. Forward for Distance PID
21. Forward for Target Distance

Remote Control

1. Minefield Challenge
2. Introduction to Remote Control
3. Real-Time Control
4. Mapping Values Part 1
5. Mapping Values Part 2
6. Time and Timers
7. Using Timers
8. Remote Control Buttons
9. Remote Start
10. Controlling The Arm Part 1
11. Controlling The Arm Part 2
12. Controlling The Arm Part 3

Sensing

1. The Grand Challenge
2. Configuring Sensors
3. Limiting the Arm Part 1
4. Limiting the Arm Part 2
5. Behaviors and Functions 1
6. Behaviors and Functions 2
7. Passing Parameters 1
8. Passing Parameters 2
9. The Ultrasonic Rangefinder
10. Forward until Near
11. Straight until Near
12. Straight until Near (Fine Tuning)
13. The Line Tracking Sensors
14. Calculating Thresholds
15. Basic Line Tracking
16. Line Track for Distance
17. Optimized Line Tracking
18. The Gyro Sensor
19. Turn for Angle Part 1
20. Turn for Angle Part 2
21. Intro to the LCD
22. Displaying Text
23. Continually Updating the Display

Engineering

Inventor's Guides

Engineering

1. Safety Considerations
2. Motion
3. Structure
4. 3-wire Motor Module
5. 2 Wire Motor 393
6. Servo Module
7. VEX Cortex Microcontroller Schematic
8. Motor Controller 29
9. 2-Wire Motor 269
10. Sensors
11. Shaft Encoders
12. Potentiometers
13. Ultrasonic Rangefinder
14. Line Follower
15. VEX LCD

Building Instructions

Setup

1. RECBOT Building Instructions (Cortex)
2. RECBOT Building Instructions (PIC)
3. Squarebot4 Building Instructions
4. Clawbot with Sensors Building Instructions
5. Base Clawbot Building Instructions
6. Swervebot Building Instructions

Challenges

Movement

1. Labarynth Challenge
2. Sumobot Challenge
3. Wait States Investigation
4. Simulated Acceleration
5. Power Levels Investigation
6. Turning Investigation
7. Sentry Simulation Level 1
8. Driving Straight Challenge
9. Basketball Drills
10. Power Level Investigation
11. Turning with Encoders Investigation
12. Driving Straight II
13. Seeing the Difference
14. Robot Acceleration / Deceleration

Remote Control

1. Minefield Challenge
2. Race to the Finish
3. Robo-Slalom Investigation
4. Round Up
5. Bull in the Ring
6. Remote Control Buttons
7. Robo-Dunk
8. RoboWriter
9. Turn Buttons

Sensing

1. Grand Challenge
2. Wall Follower
3. Robotic Mouse
4. Robocci
5. Quick Tap
6. Addition and Subtraction
7. Optimizing Code
8. Incorporating Functions
9. Real World Values
10. Seeing the Difference
11. Robo-Dunk 2
12. Robot Acceleration
13. Sentry Simulation 2
14. Speed of Sound
15. Sentry Simulation 3
16. Sonic Scanner
17. Forward until Dark
18. Table Bot
19. Robo 500
20. Robo Slalom Level 2

Quizzes

Engineering

1. Safety Attitude Test
2. Safety Quiz
3. General Safety Test
4. Robotics Lab Inspection Sheet

Assessment Rubrics

Engineering

1. Writing Criteria
2. Proposal Assessment Rubric
3. External Design Reviews
4. Workplace Competencies
5. Internal Design Reviews
6. Robotics Exploration Rubrics
7. Working Habits
8. Presentation Rubrics
9. Engineering Journal Rubric

Additional Resources

Resources

1. ROBOTC.net
2. ROBOTC Forums
3. Robotics Academy
4. CS2N

Print Materials

Fundamentals

1. Behaviors
2. Pseudocode & Flowcharts
3. Thinking About Programming
4. White Space
5. Reserved Words
6. Comments
7. ROBOTC Error Messages
8. ROBOTC Rules
9. Natural Language - VEX Cortex Reference
10. Natural Language - VEX PIC Reference
11. Sense Plan Act (SPA)
12. Boolean Logic
13. While Loops
14. Shaft Encoders
15. Line Follower
16. Servo Module
17. Behavior Based Programming
18. White Space
19. If-Else Statements
20. Ultrasonic Rangefinder
21. Light Sensor
22. VEX Claw
23. Pseudocode & Flow Charts
24. Thresholds
25. Functions
26. Touch Sensors
27. Accelerometer
28. VEX Flashlight
29. Comments
30. Variables
31. Timers
32. Potentiometers
33. VEX LED

Setup

1. VEX Cortex Driver Installation
2. Establish a VEXnet Link
3. VEXnet Joystick Calibration
4. USB to Serial Cable Driver Installation
5. Download Sample Program Over USB

Movement

1. Running a Program
2. While Loops
3. If - Else Statements
4. Variables
5. Global Variables
6. Integrated Encoder Module

Remote Control

1. Timers

Sensing

1. Servo Modules
2. Switch Cases
3. Potentiometers
4. Thresholds

Engineering

1. Safety is an Attitude
2. General Lab Safety
3. Electrical Safety
4. Power Tool Safety
5. Safety Checklist
6. VEX Microcontroller Schematic
7. Engineering Process Reference
8. Keeping and Engineering Journal
9. Design Review
10. Understanding the Problem
11. Brainstorming
12. Engineering Definitions
13. Team Building
14. Recording Progress
15. Gantt Charts
16. Preparing for a Competition
17. PERT Charts
18. Planning Your Time
19. ROBOTC Software Inspection Guide

Additional Resources at vexteacher.com

Engineering Videos

Teachers will find videos that teach about: Engineering Process, Project Planning, Flowcharts, Iterative Design, and Atlantis, a Calculated Deep Sea Adventure.

Engineering PDFs (Handouts)

There are PDFs in the project management section of the VEX Cortex Video Trainer that cover: Keeping and Engineering Journal, Team Building, Problem Solving, Time Management, Design Reviews and more. The PDFs below are designed to accompany the VEX Cortex Video Trainer and teach design and engineering.

Hand Tool Identification Guide	Technical Sketching Worksheets	Rube Goldberg Challenge Handouts	VEX Hot Dog Maker Challenge

Teaching The Engineering Design Process

“The Engineering Design Process is the formulation of a plan to help an engineer build a product with a specified performance goal.”

This process involves a number of steps, and parts of the process may need to be repeated many times before production of a final product can begin. Engineering design teaches problem solving, brainstorming, time and resource management, cooperation and collaboration, and the soft skills that today's workforce demands. Robotics and robotics competitions provide a high energy organizer to teach the Engineering Design Process.

VEX Robotics Competitions - Each year the Robotics Education and Competition (REC) Foundation hosts an International VEX competition, in 2014 there were over 10,000 teams. The Competition is incredibly well run and can provide a life-changing experience for students. The Robotics Academy is a strong supporter of the formalized robotics competitions because they are well thought out, provide real deadlines, and provide students with a high energy 21st century learning experience.

School-based Competitions - Many teachers use school based competitions. They setup classroom design challenges; some of them are competitive and others are cooperative in nature. There are several examples of possible school based cooperative design problems like The Rube Goldberg Challenge, and the Hot Dog Maker and Automated Workcell examples included in this curriculum.

The next couple of pages introduce resources that are available to teach the Engineering Design Process.

What is Engineering and Teaching the Engineering Design Process

Teaching engineering is an iterative process, just like engineering. Students are not going to learn engineering process by attempting to solve one engineering problem. They need to engineer solutions for many problems before they become good at it. There are multiple ways to introduce engineering and 5th grade teachers are going to approach the topic much differently than high school teachers. Start with small engineering projects that are a couple days in duration and introduce new organization concepts like brainstorming and ideation, cooperation and collaboration, Gantt and PERT Charts, and developing prototypes and Design Reviews as each project requires them. Always have students keep an Engineering Design Journal for every project. Below are some suggestions that others have used to introduce students to Engineering and Engineering Design Process.

1. Ask the class to discuss the difference between Engineering and Engineering Design Process. Help them to develop their own internal definition of what engineering is. Introduce students to engineering by showing the Engineering Process Video found on page 10 and provide them with the Definitions of Engineering PDF from page 12. Have them write a one-page paper describing “what is engineering”.
2. Introduce students to what the engineering process looks like by providing them with the Engineering Process PDF found on page 11.
3. Teach students to keep an Engineering Design Journal and require them to document their project by keeping all notes, sketches, and code snippets. Have them keep all of their handouts in their Engineering Design Notebook. There is a PDF describing the notebook on page 10.
4. Require students to work in teams. This is a very important skill for students to develop and it takes practice. Begin by having them review and discuss the “First Team Meeting” handout found on page 11.
5. Teach students how to brainstorm without offending each other. There is a handout on page 10 called “Brainstorming”. Give students ample opportunities to solve new problems regularly.
6. Require evidence of project planning and “Time Management” by having them develop PERT and GANTT Charts for various projects. Pass the Gantt and PERT chart handouts found on page 12.
7. Discuss how to break a project into manageable parts, assigning deliverables, and self assigning due dates. Have them watch the Project Planning Video on page 10.
8. Build prototypes and conduct design reviews. Use the design review handout on page 11.
9. Invite other adults to sit in on design reviews to review student solutions.
10. Require project documentation.
11. Iteratively test solutions and have students brainstorm how to improve them.
12. Give students the opportunity to present their solution
13. Debrief, talk about what worked, what didn’t work, and how to improve the process.
14. Give students other problems to solve - The students will improve each time they participate in the engineering design process. The first project may be a disaster for some teams. The debrief session is the most important part of the above process for new learners. Students need to recognize that this process will be with them in some form for the rest of their lives and so it is important that they become good at it.

Pictorial Sketching

ROBOTC 25

Pictorial Sketches

Isometric Pictorial: A drawing where length, width, and height are represented by lines 120 degrees apart, with all measurements on the same scale. When sketching in isometric, begin with the isometric axes (Figure 1). Next, make an isometric cube (Figure 2) with the same proportions as the object that you are sketching. Use the crating technique to complete the sketch.

Oblique Pictorial: There are two types of oblique pictorial: cabinet and cavalier. Cabinet oblique pictorial views (Figure 3) are commonly used to quickly sketch ideas. The height and length axes are at a 90° angle and the width axis is usually at 45° to horizontal. Height and length dimensions are actual size but the width dimension is divided in half. Cavalier oblique pictorial views (Figure 4) are commonly used for drawing thin objects that do not have a lot of width. The height and length axes are at a 90° angle and the width axis is usually at 45° to horizontal. Height, width, and length dimensions are actual size.

Remember

- Hold the pencil loosely
- Use developmental lines and the "crating" technique to keep your sketch properly proportioned.
- Only darken developmental lines when you are satisfied that they are correct.
- Keep your sketch neat!

Assessment Rubric

Use developmental lines	1 pt
Use object lines	1 pt
Sketch proper proportion	1 pt
Sketch correct	2 pts
Sketch neat	2 pts
TOTAL	7 A B C D

Complete your sketch here:

ROBOTC 15

The Adjustable Wrench

Directions
Pictured above is an adjustable wrench. Pictured at the right is a sequence of steps that can be used to layout the adjustable wrench in the rectangle below. Keep all developmental lines light. The darker layout lines pictured at the right are for illustration purposes only. Darken your sketch only when it is correct.

Remember

- Hold the pencil loosely
- Use developmental lines and the "crating" technique to keep your sketch properly proportioned.
- Only darken developmental lines when you are satisfied that they are correct.
- Keep your sketch neat!

Assessment Rubric

Use developmental lines	1 pt
Use object lines	1 pt
Sketch proper proportion	1 pt
Sketch correct	2 pts
Sketch neat	2 pts
TOTAL	7 A B C D

Complete your sketch here:

ROBOTC 14

The Open End Wrench

Directions
Pictured above is an open end wrench. Pictured at the right is a sequence of steps that can be used to lay out the open end wrench in the rectangle below. Keep all developmental lines light. The darker layout lines pictured at the right are for illustration purposes only. Darken your sketch only when it is correct.

Remember

- Hold the pencil loosely
- Use developmental lines and the "crating" technique to keep your sketch properly proportioned.
- Only darken developmental lines when you are satisfied that they are correct.
- Keep your sketch neat!

Assessment Rubric

Use developmental lines	1 pt
Use object lines	1 pt
Sketch proper proportion	1 pt
Sketch correct	2 pts
Sketch neat	2 pts
TOTAL	7 A B C D

Complete your sketch here:

ROBOTC 13

The Needlenose Pliers

Directions
Pictured above is a pair of needlenose pliers. Pictured at the right is a sequence of steps that can be used to lay out the needlenose pliers in the rectangle below. Keep all developmental lines light. The darker layout lines pictured at the right are for illustration purposes only. Darken your sketch only when it is correct.

Remember

- Hold the pencil loosely
- Use developmental lines and the "crating" technique to keep your sketch properly proportioned.
- Only darken developmental lines when you are satisfied that they are correct.
- Keep your sketch neat!

Assessment Rubric

Use developmental lines	1 pt
Use object lines	1 pt
Sketch proper proportion	1 pt
Sketch correct	2 pts
Sketch neat	2 pts
TOTAL	7 A B C D

Complete your sketch here:

ROBOTC 11

The Combination Square

Directions
Pictured above is a combination square. Pictured at the right is a sequence of steps that can be used to lay out the combination square in the rectangle below. Keep all developmental lines light. The darker layout lines pictured at the right are for illustration purposes only. Darken your sketch only when it is correct.

Remember

- Hold the pencil loosely
- Use developmental lines and the "crating" technique to keep your sketch properly proportioned.
- Only darken developmental lines when you are satisfied that they are correct.
- Keep your sketch neat!

Assessment Rubric

Use developmental lines	1 pt
Use object lines	1 pt
Sketch proper proportion	1 pt
Sketch correct	2 pts
Sketch neat	2 pts
TOTAL	7 A B C D

Complete your sketch here:

ROBOTC 12

The Hack Saw

Directions
Pictured above is a hand saw. Pictured at the right is a sequence of steps that can be used to lay out the hand saw in the rectangle below. Keep all developmental lines light. The darker layout lines pictured at the right are for illustration purposes only. Darken your sketch only when it is correct.

Remember

- Hold the pencil loosely
- Use developmental lines and the "crating" technique to keep your sketch properly proportioned.
- Only darken developmental lines when you are satisfied that they are correct.
- Keep your sketch neat!

Assessment Rubric

Use developmental lines	1 pt
Use object lines	1 pt
Sketch proper proportion	1 pt
Sketch correct	2 pts
Sketch neat	2 pts
TOTAL	7 A B C D

Complete your sketch here:

Precision Measurement

ROBOTC 27

Precision Measurement

PRACTICE IN MEASUREMENT
Use a caliper, micrometer, or steel rule to find the sizes of the following parts. Round all answers to the third digit of an inch. Find the corresponding distance for letters A through L.

Standard

A. _____
B. _____
C. _____
D. _____
E. _____
F. _____
G. _____
H. _____
I. _____
J. _____
K. _____
L. _____

Bearing Flut

A. _____
B. _____
C. _____
D. _____
E. _____
F. _____
G. _____
H. _____
I. _____
J. _____
K. _____
L. _____

Sketching Fasteners

ROBOTC 10

Screw Head Types

Directions
Sketch and label the appropriate views of the nuts and bolts below.

Nuts/Fasteners

- Nylon Insert Lock Nut - or self-locking nuts, eliminate the need for washers.
- Wing Nut - named for the two flat wings and are used when the nut has to be turned by the thumb and the knuckle.
- Wild Nuts - These are specialty type nuts used to attach parts to a base.
- Keep Lock Nuts - These nuts have a greater holding power and reduce assembly time.
- Coupling Nut - These nuts are used to provide clearance between parts.

Socket and Screw Head Drives

- Phillips - A Phillips head screwdriver is used to insert the screw. A phillips head drive is easier to locate than a standard drive.
- Hex - Hex drives have an external hex shape and can also take the largest amount of torque that is generated by a socket wrench.
- Socket Head Drive - can take a large amount of torque. They have an internal hex shape.
- Pinhead/Hex - This versatile drive lets you use either a Phillips or a standard drive.

Assessment Rubric

Use developmental lines	1 pt
Use object lines	1 pt
Sketch proper proportion	1 pt
Sketch correct	2 pts
Sketch neat	2 pts
TOTAL	7 A B C D

ROBOTC 10

Fasteners

What is the difference between a bolt and a screw?

What are the three properties that identify bolts?

- 1.
- 2.
- 3.

Sketch the following head types of bolts: (Front view)

When specifying a bolt or screw, what three bits of information must be included?

What does 1/4-20 mean?

Sketch the following drive head shapes: (Top view)

Describe the differences between a nylon insert locknut and a keps nut.

Describe the properties of a standoff.

VEX Robotics Competitions

VEX Robotics Competitions - Type Robotics Competitions into a search engine and you will have millions of hits. They are incredibly popular with over 30,000 US teams. The major advantages of competing in an annual robotics competition are:

- The competition game is new each year so your students won't get bored
- The rules are generally well thought out and developed by a team of experts
- There are hard deadlines that students need to meet.
- It is incredibly exciting for students to compete against other students from across the region and the world.
- There are multiple events that your team can compete in.
- Taking your school to these types of events adds prestige to your program.

The REC Foundation - Each year the Robotics Education and Competition (REC) Foundation hosts an International VEX competition, in 2012 there were over 7,000 teams. The Competition is incredibly well run and can provide a life-changing experience for students. There are competitions run in every state and in most countries in the world. To learn more go here: <http://www.roboticseducation.org/>

BEST Robotics <http://www.bestinc.org/> BEST stands for Boosting Engineering Science and Technology. It hosts an annual competition each year and has about 1,000 teams. To learn more go here: <http://www.bestinc.org/documents/Competition%20Overview.pdf>

VEX In-School Design Problems

On the next page you will find handouts that you can use to introduce these three challenges. The challenges are described below and are designed to challenge students to develop innovative solutions using the engineering design process.

Project	Time Required	Experience Level	Other Requirements	Focus/Coverage
Rube Goldberg	1 - week	Introductory		<p>Focus: Engineering process, teamwork, communication, systems concepts (input, output, subsystems, state), design</p> <p>Coverage: Non-Vex building of stationary "cells" in a system that can only perform their functions once, and do not have electronic control</p>
Hot Dog Maker Challenge	3-5 weeks	<p>Recommended: Students must know how to program with sensors</p>	<p>Required: Programming kit, various additional sensors, hardware and tools</p>	<p>Focus: Engineering process, teamwork, communication, collaboration, systems concepts, reliability</p> <p>Coverage: Inter-team collaborative design and Vex building of subsystems that must work together reliably to accomplish a multi-step process with dissimilar functions at each step. Project management on several levels is key to successfully completing this activity.</p>
Automated Work Cell	4-5 weeks	<p>Required: Students must know how to program with sensors</p>	<p>Required: Programming kit, various additional sensors, hardware and tools</p>	<p>Focus: Engineering process, teamwork, communication, collaboration, systems concepts, reliability</p> <p>Coverage: Inter-team collaborative design and Vex building of subsystems that must work together reliably to accomplish a multi-step process with dissimilar functions at each step. Project management on several levels is key to successfully completing this activity.</p>

Notes on Projects:

1. Rube Goldberg Challenge – This challenge can be accomplished with or without VEX parts. Our intention with this design problem is to get the kids engineering early and have them develop an outside the box solution. We believe that the project can be done without VEX parts using all recycled parts. The project can be done at home as a homework assignment over the weekend and then brought to class. There are many examples of Rube Goldberg Machines on the Internet. There is also a handout below which uses VEX parts.
2. The Hot Dog Maker & Automated Work Cell – The nature and themes of these projects can be modified however the teacher sees fit (hot dog maker and work cell are just two ideas), and should be designed to fit available resources. The idea behind the projects is to challenge the students to develop an automated system that requires every group to be doing something different, yet integral, to the final solution. The solution to the system is up to the students.

LESSONS ENGINEERING 101 | Rube Goldberg Challenge STUDENT

Rube Goldberg Challenge

This lab was taken from Professor Howard Choset's General Robotics class at Carnegie Mellon University. (http://generalrobotics.org)

In this lab you will design a simple Rube Goldberg Machine. The idea is that this machine should perform an overly complex process in order to complete a simple task. The images below should clarify this idea.

For more information on Rube Goldberg machines, including an annual national contest, visit the official Rube Goldberg website at: www.rgm.com

As you walk used outdoor shoe, hook (A) strikes suspended ball (B), causing it to kick football (C) through goal posts (D). Football drops into basket (E) and string (F) lets spinning can (G), causing water to soak coat into air, via coil spring, cord (I) opens door (J) of cage, allowing dog (K) to walk out on porch (L) and grab worm (M), which is attached to string (N). This pulls down window shade (O), on which is written, "YOU SAK MAH, THAT LETTER!"

Challenge Statement
Build a Rube Goldberg Machine that will dispense a ball when a quarter is inserted. Your machine should consist of at least 5 energy transfers (steps). You may generate your own electrical potential, or utilize more creative sources of same, as long as you do not use a commercial product like a battery, power supply, or outlet power). You may use any materials you can find except for those listed below. Because this machine will be unique and contribute to the group. Because this means you can't, for example, have some rolling ball hit a pin that in its way open a ramp and have those actions count as steps. If you have questions now is the time to ask.

© 2015 Carnegie Mellon Robotics Academy

ROBOTC 3

The VEX Goldberg Perpetual Motion Machine Cooperative Challenge

The spirit of Rube Goldberg, VEX Goldberg's motto, lives on in this engineering problem solving activity. Read carefully and envision your solution to solve the VEX Goldberg Perpetual Motion Engineering Challenge.

OVERVIEW
The class will be responsible for working together to design a "TheVEX Goldberg Machine". The class will be broken into multiple teams. Each team must design, fabricate, and demonstrate their part of the "VEX Goldberg Perpetual Motion Machine". Each subsystem within the larger system must integrate with the other subsystems to make up the whole system. It will go to the class to determine which subsystems to choose. Each part of the machine should consist of at least three steps. A step is a linear process, not a parallel process. If an action causes a reaction, it is an example of one complete step. If an action causes two things to happen, it would be a parallel result and would only count as one step. Your solution may use a combination of potential energy and electro-mechanical energy provided by the VEX robotics kit. Each step must be unique and contribute to the solution. Each design team must build a mechanism that is part of the larger mechanism that moves a ball from one point to another.

Evaluation is based on the following criteria:

1. The ball begins lower through the mechanism after only one human intervention.
2. The human cannot touch the mechanism once the system starts.
3. Teams are limited to one VEX kit. They may also use VEX accessories and recycled materials. Absolutely no new materials may be used to solve this problem.
4. Each team's mechanism must fit within a 3' foot cube.
5. The ball must move through the whole system back to demonstrate perpetual motion concept.

COOPERATIVE GROUPS
Students will work in teams of 3 or 4.

BEGINNING STEPS
Each group will:

1. Select a project manager
2. Brainstorm solutions to the challenge
3. Present their ideas to the whole group
4. Participate in a class discussion determining:
 - a. Order of systems
 - b. Modification of subsystems
 - c. Overall strategies to control the system
5. Design and fabricate solutions
6. Test subsystems
7. Help other groups get their subsystems operational
8. Integrate and test subsystem communication
9. Test integration of subsystems into overall system

© 2015 Carnegie Mellon Robotics Academy

ROBOTC 4

Student Rube Goldberg Challenge Notes

It is extremely difficult to get a diverse group of people to agree on many things. It becomes impossible if all members of that team want to have things their way. Below are general rules that apply to all team oriented problem solving situations. If the team would like to maintain good group dynamics, it is important that they spend several minutes talking about these important rules.

Listening tips for your team

- Listen, easy to say, hard to do. Listening is hard work, work at it
- Never assume anything.
- Don't jump to conclusions. Listen to all options before you form your own.
- Focus on the problem, it is easy to get sidetracked.
- Be positive.
- If you aren't getting it, listen harder to your teammates.
- Ask questions.

Brainstorming ideas
Students need to understand that there are no bad ideas. All ideas are to be treated with respect. Often, one idea will lead to another idea, which may seem off the wall to the group may stimulate an idea from someone else. Many people are shy and won't share their opinions, particularly if they feel they will be ridiculed.

Be flexible
There is more than one way to solve any problem. Team members must be prepared to compromise, combine good ideas, listen to others opinions, and test all suggestions fairly and with respect.

Consider resources
All problem solvers have a limited set of resources. When problem solving, your best logical resources are people and information. If you know someone with expertise in the area on which you are working, consult them. Put in time up front, researching how others have solved similar problems. There is no month to "invent the wheel". There is one resource that is your control at the beginning of the problem, but becomes totally out of your control as the project moves along. There is a saying that "time waits for no one." That becomes extremely important when you have a fixed deadline.

Test multiple solutions
Students often select the first idea that comes to mind without thinking about other options. It is important for the team to take the time to look at several options if they want to come up with the best solution.

Be positive
Do not see yourself as "part of the solution" or "part of the problem?" We all would like to see ourselves as part of the solution. Unfortunately it doesn't always work out the way that we would like and this can lead to unhappy teammates. It is imperative that you understand how important it is to be positive when you are working in a team.

VEX® Cortex™ Video Trainer Using ROBOTC™ © 2015 Carnegie Mellon Robotics Academy

ROBOTC 2

The Hot Dog Maker Challenge

The brilliant Hot Dog Shop Inc. is scouting a new outlet on a large college campus. Due to the nature of the diet of college students (hot dogs are crucial to survival), the Hot Dog Shop has experienced long lines during peak hours with some customers waiting away hungry. Investigate whether it would be cost effective to develop an automated system for serving cooked hot dogs. The objective is to serve the customer while limiting human intervention to one person. Your team's job is to develop an automated hot dog maker prototype out of VEX parts. You will present your completed prototype to our design team.

Task
Your team's objective is to develop an automated system for delivering cooked hot dogs once they are ordered. A single person, working from the point of sale, will press a button that releases an order that automatically loads and delivers a cooked hot dog. Your team will then analyze the long term benefit of implementing your system. Your analysis must weigh the pros of this approach with the cons, and make them clear to the client. Be sure to consider environmental, financial, and social factors in your analysis.

Time Frame
A draft of your proposal will be due one week from today and will be presented in class. Your team's final presentation will be given two weeks from today to an executive representing your client company.

Structure
You will select one partner from your company to work on your presentation. You may use any combination of the following formats for your formal presentation:

1. PowerPoint Presentation
2. Poster Presentation
3. Video

Rubric

A successful presenter will accomplish the following:

1. Explain the problem.
2. State the benefits to society and the company.
3. State the drawbacks of the robotics approach to society and the company.
4. Discuss what specific functions and tasks the robot would be required to accomplish.
5. Discuss cost and labor force impacts.
6. Describe your proposed solution.
7. Present an initial set of plans.
8. Present a timeline for proposed work.
9. Cite your sources.

Where to Start
Feel free to use any resources you can find, but remember to cite all sources.

VEX® Cortex™ Video Trainer Using ROBOTC™ © 2015 Carnegie Mellon Robotics Academy

ROBOTC 3

Rube Goldberg Challenge Lab Grading Sheet

Introduction to Engineering

Team _____ Final Grade _____

Member _____

Member _____

Member _____

Minimal Standards

1. Size: no bigger than 3' x 4' x 5'
2. At least two energy transfers occur
3. There can be ONE or two human interventions. One that originates with the machine, and one just in case something unusual happens.
4. This grading sheet must be filled in with team and member information and presented to the grader.

Grading
Grading only occurs if the minimal standards were met completely.

Total number of human interventions (Circle one): 1 2 3

Number of Energy Transfers (Circle one): 1 2 3 4 5

Final Grade: _____

Scale (circle and total)

5 Energy transfers: 100

4 Energy transfers: 90

3 Energy transfers: 75

2 Energy transfers: 60

+8 bonus for one connected machine (with 5 transfers): _____

+2 bonus for each additional connected machine (with 5 transfers): _____

Notes

VEX® Cortex™ Video Trainer Using ROBOTC™ © 2015 Carnegie Mellon Robotics Academy

Breaking Programs into Behaviors

What Are Behaviors?

A behavior is really anything your robot does: turning on a single motor is a behavior, moving forward is a behavior, tracking a line is a behavior, navigating a maze is a behavior. There are three main types of behaviors that we are concerned with: complex behaviors, simple behaviors, and basic behaviors.

Complex Behaviors

These are behaviors at the highest levels, such as navigating an entire maze. Though they may seem complicated, one nice property of complex behaviors is that they are always composed of smaller behaviors. This means that if you observe a complex behavior, you can always break it down into smaller and smaller behaviors until you eventually reach something you recognize.

Simple Behaviors

Simple behaviors are small, bite-size behaviors that allow your robot to perform a simple, yet significant task, like moving forward for a certain amount of time. These are perhaps the most useful behaviors to think about, because they are big enough that you can describe useful actions with them, but small enough that you can program them easily several lines of code.

```
setMotorSpeed(leftMotor, 50);
setMotorSpeed(rightMotor, 50);
sleep(2000);
```

The simple behavior directly above enables a robot to move forward for two seconds.

Basic Behaviors

At the most basic level, everything in a program must be broken down into tiny behaviors that your robot can understand and perform directly. These are behaviors the size of single lines of code, like turning on a single motor, or checking a single sensor port.

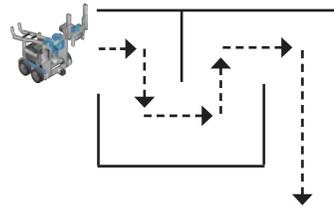
```
setMotorSpeed(leftMotor, 50);
```

Basic behavior - turn on one motor.

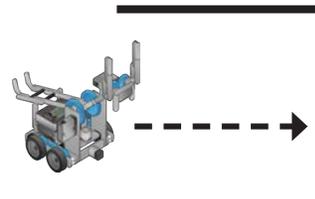
Exercises

1. What level of behaviors can your robot perform directly?
2. Why is it useful to think about a robot's actions in terms of behaviors?

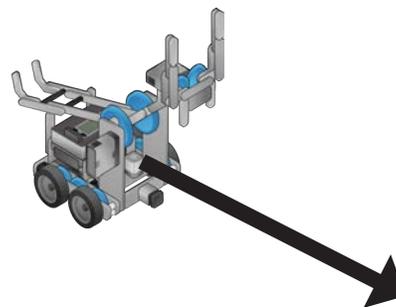
Complex Behavior:
Maze Navigation



Contains the Simple Behavior:
Move Forward until Touch Sensor is Pushed



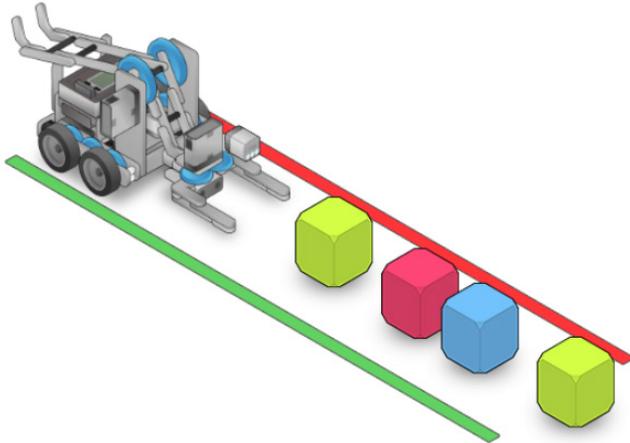
Contains the Basic Behavior:
Turn on Left Motor in Forward Direction



Sense Plan Act

How does a robot think?

One easy way to understand how a robot thinks is **Sense-Plan-Act**. A robot must be able to Sense its environment, Plan a course of action based on that data, and Act on that plan.

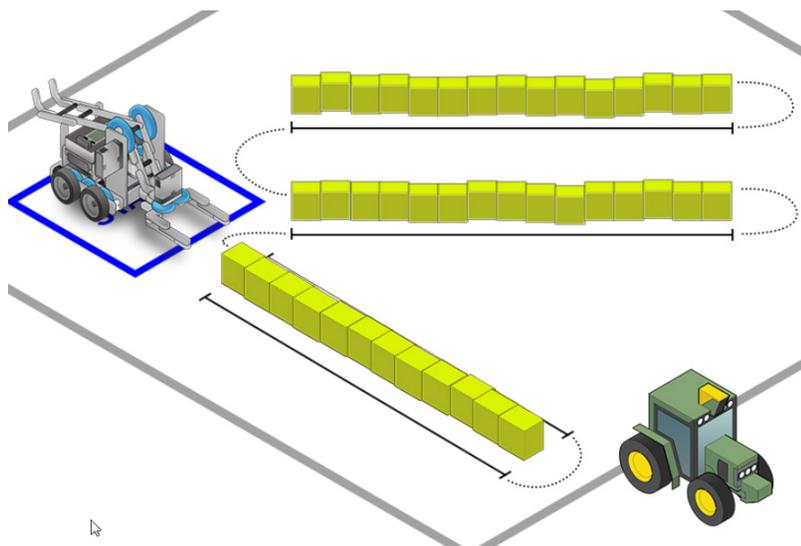


Sense

Using a variety of available sensors, the robot gathers data from its surroundings. Sensors include anything that provides the robot with information on its environment, such as the color sensor mounted on the robot in the picture, which will provide feedback about the color of the blocks in front of it.

Plan

The robot will process the information gathered in the Sense phase, and formulate an appropriate plan of action to react to what it saw. This step is most often performed by software (like your ROBOTC software) that has been loaded onto the robot in advance. The program illustrated here tells the robot to go forward until it sees a color.



Act

The robot acts in the world through the use of actuators—any component which allows the robot to create a change in its surroundings, such as motors, which move the robot through the environment. The robot in the picture will drive through the maze.

Answer the following questions

1. Define what a robot does.
2. Describe how your robot senses, plans, and acts to solve the challenge that you are working on.

Teaching Pseudocode and Flowcharts

Introduction

Pseudocode - is a native language description of what the robot is required to do. With practice pseudocode eventually resembles ROBOTC code.

Flowchart - is a graphical representation of program flow.

It is a very good practice to have students begin each programming task by breaking the task into its smallest parts and develop pseudocode that describes the robot's behaviors. You will find much more detail on this practice in the *Introduction to Pseudocode* lesson.

The practice of developing pseudocode is informally introduced on the first day of class when students are asked to write a program to control a humanoid robot to make a sandwich. See *Introduction to Robot Programming* for a full description of that activity. In that activity, students will not yet know a robot programming language, but they can begin to write pseudocode to describe the behaviors the robot will need to complete to make the sandwich. (E.g., lift the arm, open the hand, reach for the bread, grab the bread, etc.).

In this teacher's guide, we use the term *flowchart* in a very general way to convey that students are constructing visual representations of the problem and its solution. The advantage to flowcharts is that they help a student identify robot decision making.

This curriculum doesn't formally introduce flowcharts until the Remote Control Chapter because the first chapter, Movement, only require simple behaviors (movingForward, turning, moveUntil an encoder count). During these activities, students will learn the basic lexicon of the programming language allowing them to begin writing robot behaviors using pseudocode. Teachers should encourage students to use pseudocode - even if it initially starts as common descriptions and slowly improves to reflect a more accurate programming language. Pseudocode will eventually be used to describe individual parts within a flowchart since each pseudocode phrase describes a unique robot behavior.

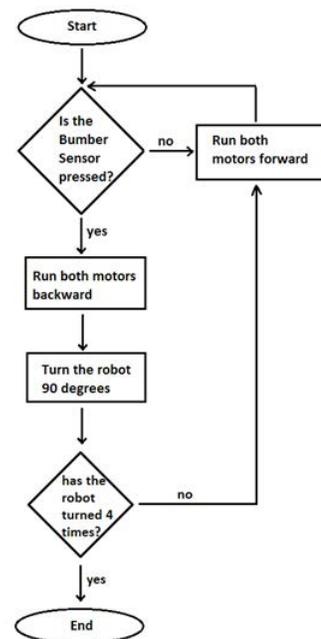
In the Remote Control Chapter, students will begin programming their robots to solve problems that contain multiple steps with decisions, that is when this curriculum formally introduce flowcharts. With practice, students will begin to see that flowcharts provide a visual representation of the robot's decision making process and then begin using flowcharts to solve their programming challenges.

Example pseudocode

```

Solve the Maze
Move forward ---> move forward 60 cm
turn left ---> turn left 90 degrees
move forward ---> move forward 80 cm
turn right ---> turn right 90 degrees
move forward ---> move forward 60 cm
turn right ---> turn right 90 degrees
move forward ---> move forward 50 cm
  
```

Example flowchart



Introduction to Pseudocode

Overview

In order to plan a program and write efficient code students need to be able to write clear instructions for the robot. This lesson introduces students to the idea of writing clear instructions and then introduces them to pseudocode.

Objectives

Students will be able to:

- Listen carefully and follow instructions
- Communicate clear instructions
- Break tasks down into smaller pieces
- Write pseudocode for a simple maze
- Understand the necessity of planning clear steps

Materials

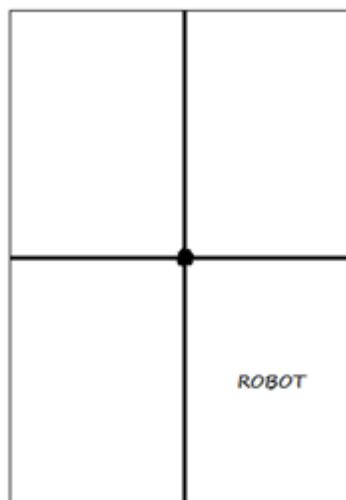
- Blank paper for each student
- Pencil and a ruler for each student
- A large table top or floor surface to setup a simple maze
- Tape to mark out maze boundaries
- Small box or object to represent a robot

Clear Communications Lesson

Procedure

1. Introduce the following “Clear Communications” drawing activity. Tell students:

Robots will follow the program that they are given, even if that program doesn't make sense. The robot needs to be given a program that it can understand to produce the desired outcome. Let's begin with a drawing activity to help us think about this idea of giving clear instructions.



Have all students begin with a blank sheet of paper and then give them the following verbal instructions:

1. *Draw a dot in the center of your page*
2. *Draw a vertical line from the top of your page to the bottom of your page, passing through the center dot*
3. *Draw a horizontal line from the left of your page to the right of your page, passing through the center dot*
4. *Write the word ROBOT center of the square created at the bottom right of your page*

Have students hold up their drawings when everyone is finished. Check to see that everyone's is the same. Discuss any differences if they arise.

Introduction to Pseudocode

Instruct students to:

Write down the behavior "Solve the maze" on the top of your paper.

Now, tell them to:

Break down that behavior into smaller behaviors and write them below in order.

If students are unsure, ask:

What does the robot need to do to follow the solution path?

(Move forward, turn left, move forward, turn right, etc.)

Ask students:

Is your list of behaviors is clear enough to instruct the robot through the maze.

They should realize that we are close, but the robot doesn't know how far to move forward each time or how much to turn.

Finally, instruct students to:

Use your rulers to figure out the distance the robot needs to move for each behavior.

(Exact distances will depend on your maze setup).

Write this new specific behavior next to their last list of behaviors.

We now have a list of behaviors specific enough to give as instructions to the robot.

Tell students:

By starting with a very large solution behavior and breaking it down into smaller and smaller sub-behaviors, you have a logical way to figure out what a robot needs to do to accomplish its task.

Talking about and writing the code in English is the first step in good pseudocode practice, which allows us to plan robot behaviors before we translate them to code.

Example Pseudocode

Solve the Maze

```

Move forward ---> move forward 60 cm
turn left ---> turn left 90 degrees
move forward ---> move forward 80 cm
turn right ---> turn right 90 degrees
move forward ---> move forward 60 cm
turn right ---> turn right 90 degrees
move forward ---> move forward 50 cm
  
```

Pseudocode Exercise

What is Pseudocode?

Robots need very detailed and organized instructions in order to perform their tasks. Before a programmer can begin programming they need to break a robot's behaviors down into simple behaviors and figure out when each behavior should run. Some programmers like to use pseudocode to begin constructing the programming problem.

pseudo

adj : not genuine but having the appearance of;

Source: WordNet ® 1.6, © 1997 Princeton University

Pseudocode is a hybrid language, halfway between English and code. It is not real code yet, but captures the details that will be important in translating your ideas to code, while still allowing you to think and explain things in plain language. Good pseudocode will make it very straightforward to write real code afterwards, because all the behaviors and logic will already be contained in the pseudocode.

Pseudocode example

If you wanted to program a robot to stop when it saw an object and move forward when it didn't see an object your pseudocode might look like:

pseudocode

1. Move forward
2. If (sonar sensor detects an object)
 - stop
3. When the sonar sensor no longer sees an object move forward.
4. Do this forever

Exercise

1. Convert these instructions to pseudocode and into a flowchart:
 - a. "If it's raining, bring an umbrella."
 - b. "Keep looking until you find it."
 - c. "Take twenty paces, then turn and shoot."
 - d. "Go forward until the touch sensor (on port 1) is pressed in."
 - e. "Turn on oven. Cook the turkey for 4 hours or until meat thermometer reaches 180 degrees."
 - f. "Crossing the street" Hint, make sure that you look both ways!
2. Compare the advantages and disadvantages of flowcharts and pseudocode. Explain in your own words why you believe one is better than the other. Is one of them always better than the other, or are both good in different situations? Can you use both to help solve the same problem? Should you?

Introduction to Flowcharts Page 1 of 3

Overview

A flowchart is a visual representation of program flow and is used by programmers to break down and model robot behaviors. This lesson provides teachers with a guide to introduce flowcharts in the *Remote Control Chapter*, and then this lesson should be applied to all of the subsequent Units. Students have already been introduced to pseudocode and program flow in the Introduction to Robot Programming activity; this lesson focuses on how to graphically describe robot decision making. In this lesson the teacher will model how integrate pseudocode and decision making into a flowchart.

Objectives

Students will be able to:

- Break down a problem into simple components
- Organize the components into a proper sequence
- Represent a sequence of behaviors in a flowchart
- Use a flowchart to analyze robot behaviors

For this lesson, you will need to have programmed a robot that moves forward until the touch sensor is pressed and have it ready to demonstrate to students.

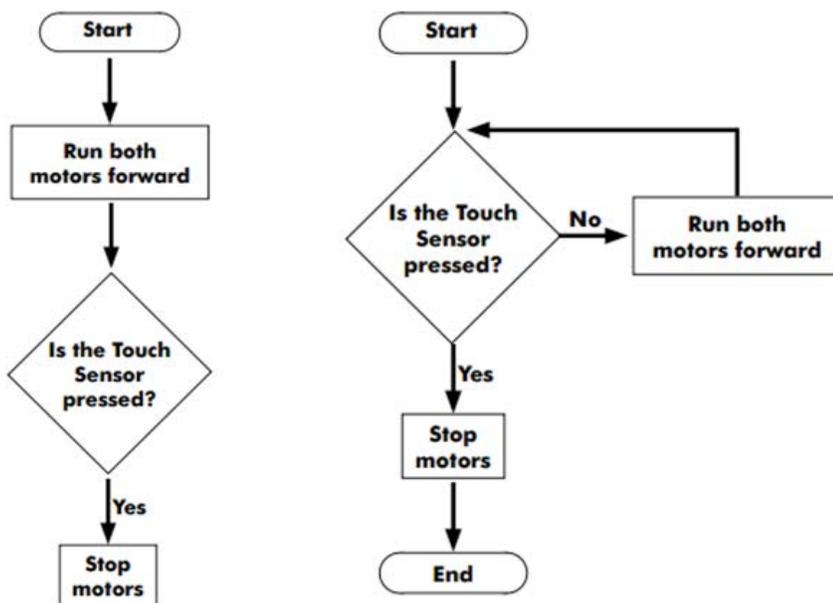
Procedure:

In the example below students will see a program that requires the robot to make a decision and includes multiple steps. Note: Show all students the picture of the flowchart 1 and flowchart 2 below.

Tell students: *Now that the robot is using a sensor to make decisions, we can use a flowchart to understand how our robot's behavior is broken down into steps within a program. So far, our flowcharts have just been single steps of pseudocode, but now we will need to add decision blocks which ask a "yes or no" question.*

Show both of the following flowcharts to the class and ask:

Which flowchart best represents a robot that needs to stop when the touch sensor is pressed?



Flowchart 1

Flowchart 2

Draw the example flowcharts on the left on the board or project them onto the screen for students to see and discuss.

Introduction to Flowcharts Page 2 of 3

If students struggle or disagree, ask:

When does the robot stop moving forward? (When the touch sensor is pressed in.)

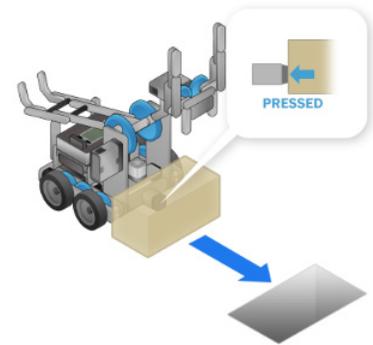
*When does the robot check the touch sensor to make the decision to move forward or stop?
(The robot continually decides to move forward when the touch sensor is not pushed, and decides to stop when the touch sensor is pushed)*

*Which flowchart represents a robot continually making decisions based on the bumper sensor?
(The flowchart on the right requires the robot to keep asking if the touch sensor is pressed)*

Read through Flowchart 2 (on the previous page) to the class to show how each robot action is represented in the flowchart. Teach students the difference between start/stop, action, and decision blocks. Tell students:

The robot must continually check the value of the touch sensor to decide what to do. If the touch sensor is not pressed, the robot continues running both motors forward. If the touch sensor is pressed, the robot stops all motors. The flow of these decisions is given in the flowchart 2 on the right, but missing from flowchart 2 on the left.

Imagine a scenario where the robot's touch sensor is pushed in while it is pushing a box off of the table. (Pictured at the right) When the box is pushed off the table, the robot's touch sensor is no longer pressed.



Refer to flowchart 2 as the class thinks about the following question:

What happens if you start the program with the touch sensor pressed in?"

Give them a moment to think, and then ask students to explain their answer to the class using a flowchart. Tell the students: *Use a flowchart to break down each step of the robot's behavior so that we can see what actions the robot will perform in any situation.*

See the flowchart 3 at the right. Have students copy it onto their papers and fill in the blanks to properly represent program flow while a robot is pushing a box until its touch sensor is released.

If students struggle, ask:

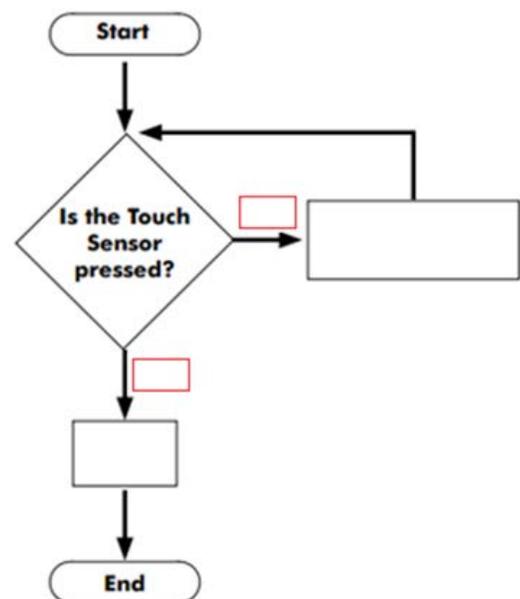
*Now when will the robot stop moving forward?
(When the touch sensor is released)*

When students have successfully filled out their flowcharts, ask the class:

What would happen if the box was not all the way against the touch sensor when the program started?

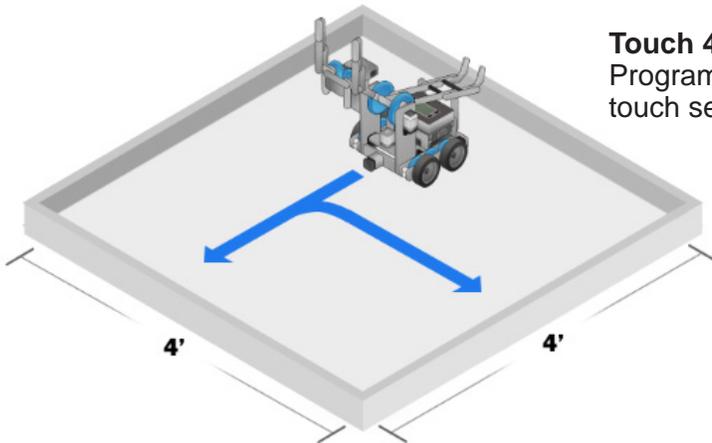
(The robot would not move)

Use a flowchart to show this pathway.



Flowchart 3

Introduction to Flowcharts



Touch 4 Walls Challenge

Program the robot to touch all four walls using the touch sensor to know when it has reached a wall

See the challenge pictured above. On a new piece of paper, have students design a flowchart to represent the robot's behavior in this challenge. The robot will need to perform more actions after the bumper sensor is read each time.

Give the students several minutes to work on their flowchart. Have them work in small groups if necessary.

If students struggle, ask:

What does the robot need to do after the bumper sensor is pressed? (Backup and turn towards another wall)

Does the robot ever need to repeat its behaviors?

(Yes, it will have the same behavior for all 4 walls)

How will the robot know when it has contacted all the walls? (When it has repeated its behavior 4 times)

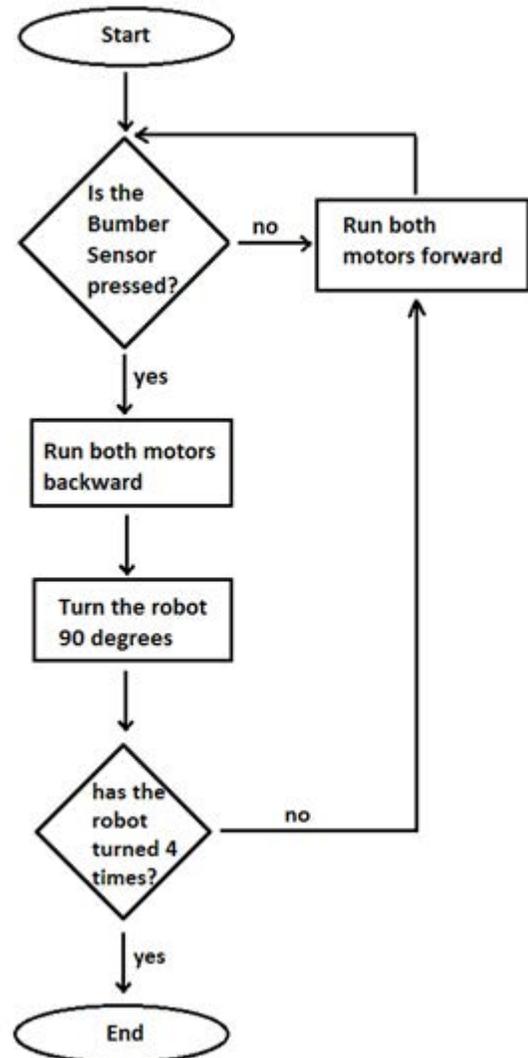
After students have finished, ask a group to draw their solution for the class. Follow the flow and check the logic for any missing steps or problem areas.

Ask if other students created a different flowchart they would like to share.

Discuss aspects of each solution until the class comes to an agreement on a correct flowchart. It's possible to have multiple correct solutions.

Tell students:

Flowcharts help us understand the decision making process of a robot. The answer to a "yes or no" decision will determine what action the robot does. By creating a properly organized flowchart, we can see the plan a robot needs to follow to be successful.



Pictured above is the solution for the Touch 4 Walls Challenge

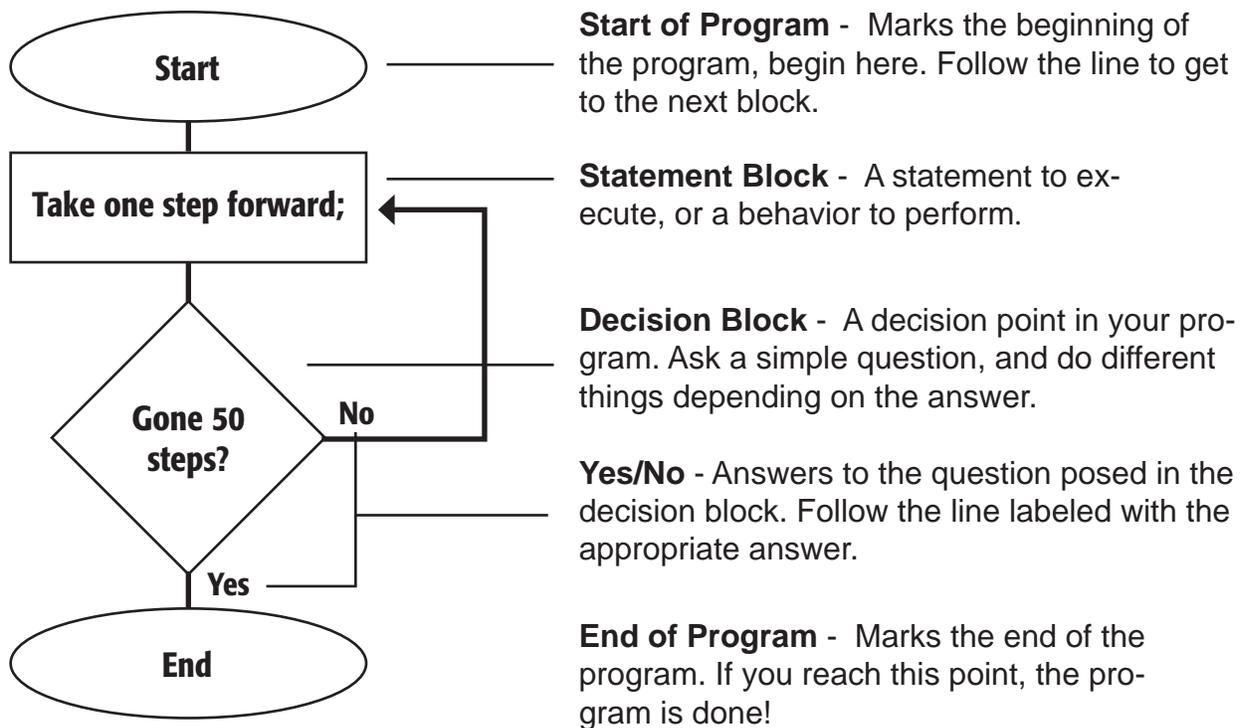
Flowchart Exercise

What are Flowcharts?

Robots need very detailed and organized instructions in order to perform their tasks. The programmer must break things down into simple behaviors and figure out when each behavior should run. A flowchart is a tool that can be used by programmers to determine program flow.

A flowchart provides a way of visually representing and organizing individual behaviors and decisions within a program -- it provides a diagram of the “flow” of the program. Programmers use flowcharts to lay out the steps that will be needed in their final program, and to help determine how the robot’s behaviors should be broken down.

Parts of a Flowchart



Exercise

1. Make a flowchart organizing the “flow” of getting ready to go to school in the morning. Be sure to include the following steps in your chart, but don’t be afraid to add other things if you need them!

Select something to wear
Take a shower
Eat breakfast
Walk or get a ride to school
Get out of bed

Look for your shoes
Brush your teeth
Put toast in the toaster
Check your alarm clock
Turn on shower

Put your shoes on
Hit snooze button
Get dressed
Comb your hair
Check the time

Introduction to Robot Programming

Programming is Precise

One of the big ideas of the Movement Chapter is that “programming is precise”. It will be important for students to change the way that they think about providing directions. The following activity will highlight the attention to detail required for students to become successful programmers.

The Humanoid Sandwich Programming Challenge

This activity requires students to program a human robot to make a two ingredient sandwich. This activity provides students with an opportunity to practice breaking down what appears to be a very simple task into its smallest parts. Students will quickly learn that they need to use very precise commands when programming robots. Making a two-ingredient sandwich (plus bread) is presumably familiar to most students, and that experience will allow them to think critically about the behaviors needed to program a humanoid robot. Initially, the majority of students will not be prepared to break this task down to a level of detail necessary for robot programming.

Procedure

The teacher says:

Your task is to program a humanoid robot to make a two ingredient sandwich. Begin by making a list of behaviors your robot needs to perform make a sandwich, then turn that list into pseudocode. Pseudocode is a simple set of instructions that you want the robot to execute. We will test your “code” by programming a human to execute your sandwich making program.

Students can begin writing their program individually or in small groups.

Identify one student to assume the role of the robot, they will stand in a place where everyone can see her/him. The props for making the sandwich can be actual bread, spreads/meats, and a utensil - or they can be stand-ins (paper for bread and/or meats, a pencil for a knife, cups for jars of spreads). The “robot” (student) will do *exactly* what fellow students say to do.

The teacher can ask the class for the first behavior to be programmed, followed by the second, third, etc. Most times, it quickly becomes apparent that students have not fully considered the level of detail required for programming. For example, the command “pick up a slice of bread” is inadequate. The robot needs to know in which direction to move to get the bread, how to detect the bread (use of sensory data), how to pick it up, etc. Those are each individual lines of code. It is not important to continue this exercise once students recognize why greater detail is required - even for a task as simple as making a sandwich.

Additional Lessons

This teacher’s guide provides multiple additional lessons that the teacher can use to introduce students to pseudocode and flowcharts. It is up to the discretion of the teacher to use or not use these lessons. For additional lessons go to the Index and find:

- Pseudocode and Flowcharts
- Introduction to Pseudocode
- Introduction to Flowcharts

Teaching How to Troubleshoot Programs

Stop, Trace, Analyze, Revise (STAR)

All teachers want students to learn how to troubleshoot their robot problems. There are times when a teacher will quickly give a student the answer, but this doesn't teach them how to troubleshoot their own programming problems, it teaches dependency. If a teacher immediately answers all student questions, the student quickly learns that all they have to do is ask the teacher and their problem will be solved. We recommend the STAR approach to teaching troubleshooting, this approach is designed to teach students stop and trace the program flow, analyze where there is a difference between what should happen and what happened, and then revise and test the program.

Teacher's Role in STAR Troubleshooting

Stop and Determine Student Intent The teacher needs to determine what does the student think is happening in the program. The teacher needs to ask the students to:

"Please describe step-by-step what is happening in your program".

Trace Together, the student and teacher, need to trace through the program and identify where the robot's behavior diverges from the student's intent.

Analyze The teacher needs to help the student to analyze what they misunderstood.

Revise Help the student to correct their misunderstanding and to fix their program. Often, correcting a student's misunderstanding means re-framing the problem or highlighting some discrepancy in how the student sees the problem.

STAR - Student's Role in Troubleshooting Programs

Stop and Reflect What do I think should be happening in each step of my program?

Trace I need to trace through the program step-by-step and identify what my program is telling the robot to do. If it is a complex program, then I should develop a flowchart that allows me to see my program flow.

Analyze If the robot is misbehaving, I need to analyze what parts of the program works and identify the point where the robot stopped doing what I wanted it to do. Then I need to figure out what I need to change to fix the problem

Revise I need to scientifically correct the program one step at a time and test each part of the program and fix the problem.

Note: Teachers might choose to post the student version of STAR somewhere in their classrooms and direct students' attention to it as they begin asking for assistance.