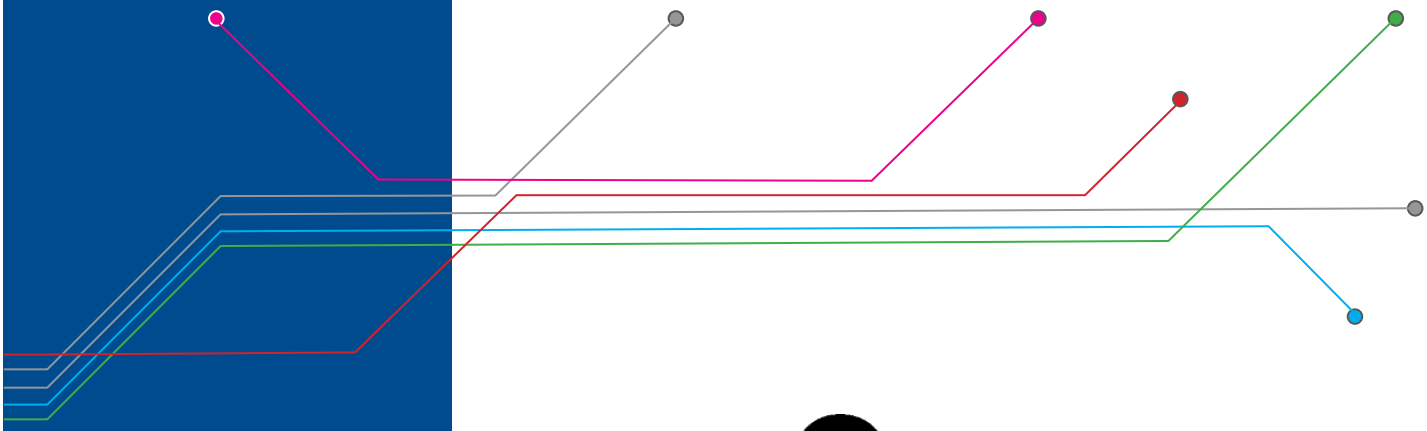
ROBOTC: Programming for All Ages



RoboMatter INCORPORATED

ROBOTC: Programming for All Ages

IDEA IN BRIEF

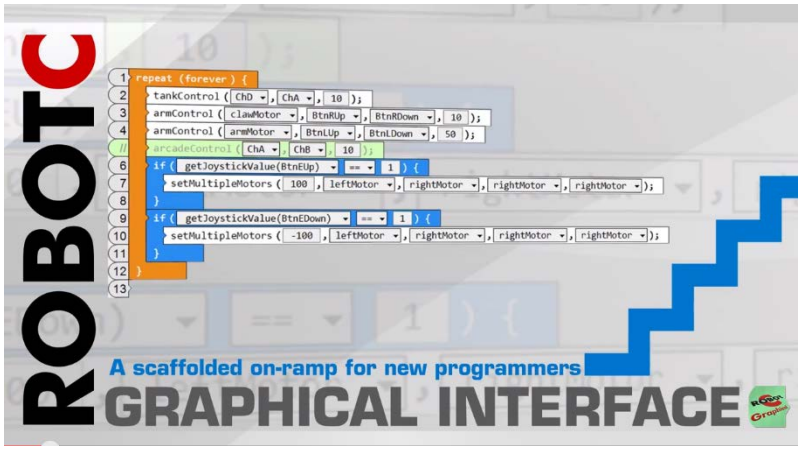
Schools often wonder which programming language or software to adopt in order to have their students develop advanced, marketable programming experience. One of the biggest concerns is the transition from simpler (novice-targeted) to more complex (professional) languages. ROBOTC is unique in offering three different forms of its programming language in order to foster those transitions as students develop increasingly sophisticated skills. Its low-entry graphical form allows novices to focus on the fundamental concepts (e.g., sequences, systems thinking, etc.); whereas the full-text form allows more advanced students to program like professionals.

ROBOTC is a C-based, robot-agnostic programming language with a Windows environment for writing and debugging programs. Although courses could be designed solely around ROBOTC, Robomatter Incorporated develops its robotics-focused courses to utilize ROBOTC as a tool to be used in activities and projects that can both introduce and deepen students' programming skills, computer science concepts, computational and algorithmic problem solving abilities, as well as other skills. Robomatter's internal development of ROBOTC as a tool guarantees not only control over the interface and its features, but also that its curriculum will remain up-to-date. New features within ROBOTC can be anticipated and the curriculum can be updated concurrently to offer schools the most recent versions of both, should the need arise.

As a cross-platform solution, ROBOTC allows students to learn the type of C-based programming used in advanced educational and professional applications. It was developed in collaboration with Carnegie Mellon's Robotics Academy and launched in 2006. With subsequent developments, ROBOTC was enhanced with the capability to program both physical and virtual robots, by adding Robot Virtual Worlds (RVW). Through ROBOTC, RVW allows students to test and run their programs virtually, and also offers different types of game-based robotics learning. ROBOTC is offered in a graphical form (block-based; i.e., ROBOTC Graphical), a text-based form with natural language (ROBOTC Natural Language), and a full-text form (ROBOTC) as an advanced source code editor using industry standard C-programming language.

Each form of ROBOTC allows its learners to succeed in programming because they are approaching programming with a tool that is accessible to them (within their zone of proximal development). The interfaces of the different forms of ROBOTC were designed to be as user-friendly to a young learner as they are to an advanced programmer, and all forms include the easiest editor possible along with an easy-to-use compiler which is capable of detecting and drawing attention to errors before

sending the program to the robot. All forms of ROBOTC also have sample programs that are available to help students to get started more easily. ROBOTC Graphical was developed because many novice programmers struggle with the syntax required within more traditional programming. By allowing students to use prepared blocks of code, students can focus on the commands within the program and its overall structure rather than on the syntax of each command or group of commands. Additionally, the application's toolbar is mostly icon-based so that students can easily find the controls that they need to use, like those for debugging, compiling, etc. There is even a converter that can show students what their programs would look like if written in text instead of the graphical language. That converter should help prepare students for writing text-based programs. ROBOTC Natural Language's text-based form allows students to become familiar with text-based programming but do so with the ease of natural language. There are debugging tools, a function library, and an automatic format fixing tool (magic wand) for building more successful programs. The compiler's capacity for detecting errors in the program is especially useful at this level when students' programs can often contain misspellings and typos. ROBOTC, the full-text form, takes students' programming experiences further and offers them three levels: Basic, Expert, and Super User. Students, or users, write syntax-dependent code and thereby program as professionals do.



By having these multiple forms (graphical, natural language, and full-text) and applications (physical and virtual), ROBOTC can be adopted for classrooms with wide ranges of students and experience levels. The question is whether it is utilized as broadly as its capabilities allow. In order to answer that question, Robomatter Incorporated visited with teachers in three different settings to find out how they are using ROBOTC.

Three Different Levels of Classrooms

Sales records for ROBOTC can only reveal so much about how it is being used. Competition teams and camps, as well as schools, purchase ROBOTC licenses for their students but likely all have different intentions for doing so. Because ROBOTC is designed to be educational as well as functional, Robomatter set out to investigate how and specifically why, ROBOTC is being adopted in educational settings. Teachers from local elementary/middle schools, high schools, and colleges were surveyed on how and why they use ROBOTC.

A programming language affords learners with a great variety of educational experiences. Which are particularly meaningful to teachers? How do the intentions of elementary school teachers using ROBOTC differ from those of college professors, or are they similar? Case studies of classes at elementary and middle school, high school, and university levels were conducted to determine how and why ROBOTC enhances these different classroom experiences.

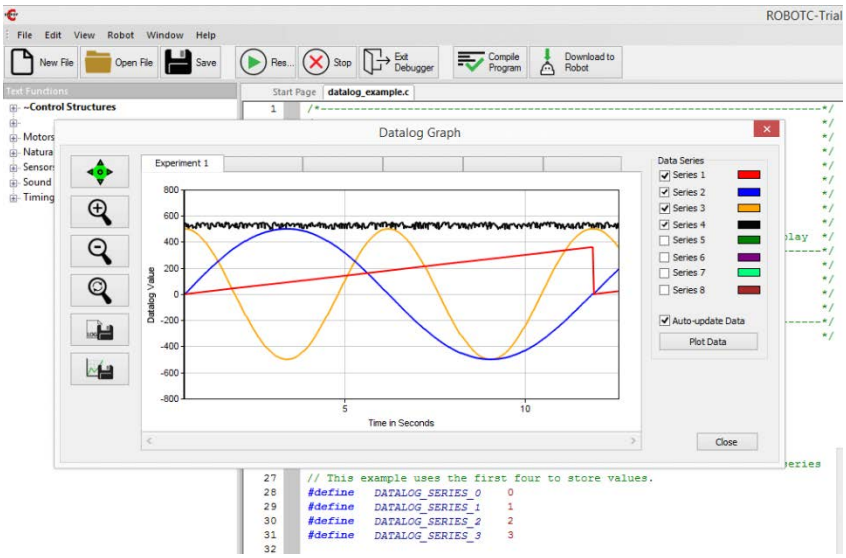
How are students using ROBOTC?

As a broadly accessible programming language, ROBOTC affords users with many opportunities to apply computational, algorithmic, mathematical, and scientific skills in the interests of programming. The teachers interviewed all expressed a deep commitment to using ROBOTC for those very reasons: 1) Students with varying levels of experience can use ROBOTC effectively because of the different forms of ROBOTC offered and because of the sample programs and templates provided within the program, and 2) Programming in ROBOTC allows students at different academic levels to focus on their applications of skills – the more important part of the class - rather than on the challenges of programming that are found with other programming languages.

ROBOTC allows classes to:	Elementary / Middle	High School	College
Use breakout boards to engineer their own robots			
Solve truly professional-level problems			
Develop function libraries			
Recognize ROBOTC as an IDE			
Consider real-world science			
Apply mathematics			
Simplify program with pseudocode in comments			

Practice algorithmic thinking			
Develop computational thinking skills			
Develop/deepen programming skills			
Develop/deepen computer science knowledge			

At the elementary and middle school level, ROBOTC is used to introduce, apply, and deepen computer science content (e.g., coding, cybersecurity, etc.), computational thinking skills (e.g., decomposition, abstraction, etc.), algorithmic thinking, and the basic programming skills like sequencing and debugging that are important to students' future academic pursuits. The use of pseudocode in comments within ROBOTC is recognized by teachers as essential to students' development of competencies – something echoed at the high school level as well. Additionally, the calculations required for programming in ROBOTC necessitate applying mathematic skills and concepts like, but not limited to: geometric shapes, circumference, proportional reasoning, distance conversions, and angles and their degrees. ROBOTC also allows teachers to introduce real-world science. For example, students learn to collect, track, and apply data through the use of sensors and data logging in ROBOTC.



At the high school level, ROBOTC allows students to apply and deepen their foundational programming skills (e.g., looped behaviors, conditional statements, variables, etc.) and their computational and algorithmic thinking skills to solve more complex problems through the use of flowcharts, pseudocode, and comments within their programs. They might also discuss software applications like ROBOTC as more than merely a programming language but also as an integrated

development environment (IDE) like those used by software developers to write and manage their source code. They also learn to use functions and the function libraries. One teacher explained that because ROBOTC harnesses the full power of C programming, "the sky is the limit" when it comes to programming and creating computational artifacts. That potential is being fully realized in university classrooms when students are better equipped to develop more demanding projects.

At the college level, university classrooms, at Carnegie Mellon University in this case, are using it for creatively demanding projects like tele-operated Urban Search and Rescue challenges and other professional-level projects that require autonomous robots. Because the complexity of the projects at the college level are similar if not identical to those faced by professional engineers, the math required for programming these projects is also sophisticated. While working in ROBOTC to complete their projects, students rely on applications of their calculus-based math skills (e.g., disderivatives crete, integrals, the Runge-Kutta method, etc.). During the course, students use ROBOTC to show proficiencies in many different robotics programming skills including but not limited to the following: wheel-free locomotion (e.g., ladder climbing), line-following, odometry/dead reckoning, path planning (waypoints or wavefront), control of an inverted pendulum robot via a proportional integral derivative controller (PID), localization by using the Kalman filter, kinematics of a robot arm, etc.

Forms of Programming and Types of Robots

Most often, students at the elementary and middle school level are using ROBOTC Graphical to learn to code both physical and virtual robots. One teacher told us that her students use ROBOTC Graphical to program both virtual and physical robots during each project because doing so allows students to test their programs virtually before running them physically. That encourages students to differentiate when a bug (problem) is in the program and when it might be in the hardware (the build). It also allows students to test their programs in a low-stakes context because students are often fearful of risking failure in front of their classmates; so, Robot Virtual Worlds (RVW) allows students to test out their programs before classmates can see. Additionally, more advanced students might begin the transition to ROBOTC Natural Language as they near the end of middle school.

Students at the high school level are most often using either the ROBOTC Natural Language or ROBOTC (full-text). However, in some cases, students who are new to programming are starting with the graphical form as part of an introductory set of lessons or courses. Students who continue through their last years often make the transition to ROBOTC (full-text) after they have advanced levels of programming skills. High school teachers report that the transition is facilitated by the hover-over and autocomplete features available in ROBOTC for assisting students. High school teachers generally report using ROBOTC to program both the physical and virtual robots for the same reasons as elementary and middle school teachers do. However, some have added that students programming with ROBOTC in RVW can accomplish more in less time. Also, RVW allows students to develop their projects in parallel to other students who are developing physically-based projects. This is especially helpful in schools where resources might be limited.

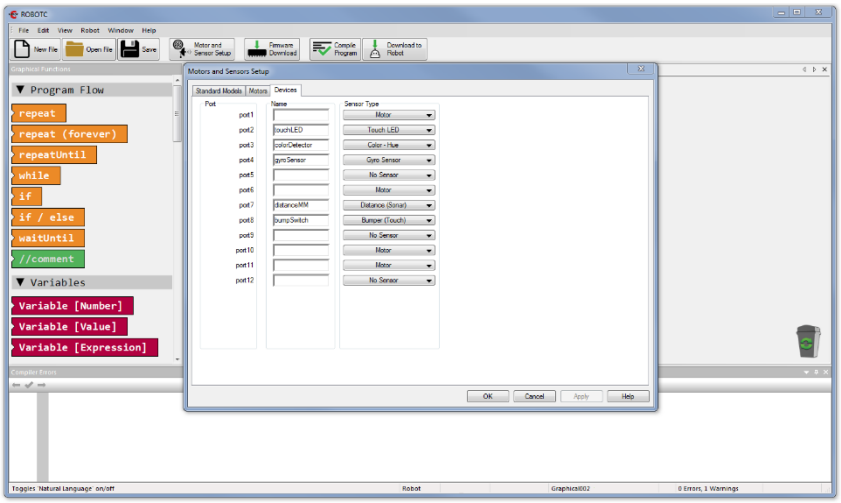
Students at the college level who are applying calculus to their solutions are most often using ROBOTC (full-text); however, introductory robotics courses at the college level sometimes start with ROBOTC Natural Language, or even ROBOTC Graphical for true novices to programming. Although some college classrooms might include some programming in ROBOTC for RVW, most of the programming in ROBOTC is for physical robots. In fact, some college classrooms are using ROBOTC to program breakout boards so that students can also design their own hardware.

Forms of ROBOTC used:	Novices	Experienced	Advanced
ROBOTC Graphical			
ROBOTC Natural Language			
ROBOTC (Full-Text)			

Connections between Programs and Hardware

An important feature of ROBOTC is helping students to recognize how the programming affects the robot's behaviors. This is especially important for younger learners and novice programmers and in fact, the teachers at the elementary, middle, and high school levels reported that one of the most

important features of ROBOTC was its ability to highlight the relationships between programming and hardware behavior. Starting with the setup of the motors and sensors in ROBOTC, teachers remarked that students more readily recognized how input/output were related. The real-time feedback and debugging afforded by ROBOTC further highlighted the connections between programs and hardware in these classrooms. Because ROBOTC is a realistic programming language, even in graphical form, teachers reported that students were able to see the connections of their programs to their robots, their devices, and to the Internet of Things (IoT).



Presumably, most college students recognize the connections between their programs and hardware. Teachers at this level did not report the need for ROBOTC to highlight those connections but it seems reasonable to assume that novices to programming might be benefiting from the same ROBOTC features (hardware setups, feedback, debugging, etc.) that younger students do. However, because the projects required at the college level are much more challenging and often require students to engineer their own hardware, those connections are prerequisite to effectively solving challenges like those within ping-pong duels, robot tag, stair climbing, etc.

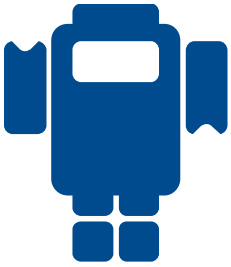
Development of Generally Useful Skills

One important commonality across the three levels of classrooms using ROBOTC was that teachers all recognized ROBOTC, and the activities contextualized in and around ROBOTC, as valuable opportunities for students to develop skills that are related to, but not exclusive to, robotics. Teachers reported that they were using ROBOTC, in part, to help students to build computational thinking skills (e.g., decomposition, sequencing, debugging, systems thinking, etc.). Additionally, they reportedly value ROBOTC-centered activities and projects for being realistic and complex because those projects necessitate that students learn to persevere in the face of challenges, and to effectively collaborate. These were identified by teachers as generally useful skills

that teachers wanted students to develop in order to prepare them, not only for subsequent robotics or computer science courses but also for productive, successful futures.

Summary

ROBOTC is being effectively used to engage, challenge, and educate students who range in ages and experience levels from the elementary-school level to the college level, and from novice to advanced. Its availability in three different forms (graphical, text-based with natural language, and full-text) allow ROBOTC to afford teachers and students with ample opportunities to learn how to effectively program both physical and virtual robots, to develop deep computational thinking skills, to solve complex real-world problems, and to progress from block-based to full-text programming as the students develop more sophisticated skills. As one teacher simply stated, “With the full power of C [programming language underlying ROBOTC], the sky is the limit!”



About Robomatter

Robomatter Incorporated is a technology and engineering curriculum solutions provider focused on bringing STEM experiences to students by leveraging the motivational effects of computer science and robotics. Robomatter's solutions include research-backed curricula, interactive technology tools, STEM teacher professional development programs, and certification programs through a partnership with Carnegie Mellon's Robotics Academy.