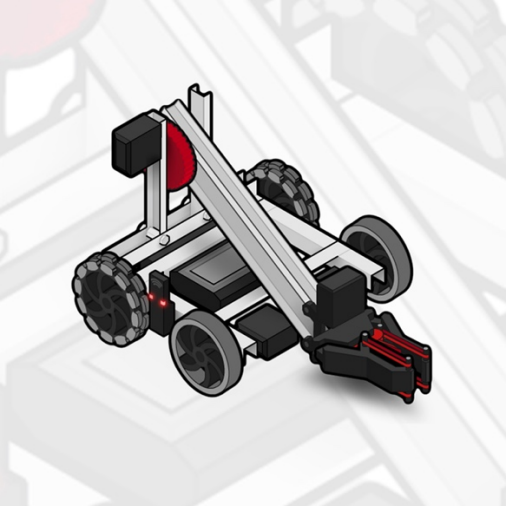
Standards Alignment Guide



Coding and Computational Thinking with VEX V5 Curriculum Standards Alignment Guide

Computer Science Teachers Association (CSTA), Computer Science Principles (CSP), Next Generation Science Standards (NGSS), Common Core Mathematics, Common Core English Language Arts, International Society for Technology in Education (ISTE)

Last Updated: 11/2021

Carnegie Mellon Robotics Academy

The Standards

These standards are aligned with the instruction in Coding and Computational Thinking with VEX V5 to ensure that all students are given clear and precise instructions and that teaching practices focus on the learning objectives. These standards can be used to track student performance and to set expectations for student achievement.

The Big Ideas

Robotics can be something you teach with, as well as something you teach about. Coding and Computational Thinking with VEX V5 uses robots, and covers robotics content, but ultimately seeks to give students experience and access to a much broader set of skills and perspectives of Computational Thinking.

1. Role of the Programmer
2. Sense, Perceive, Plan, Act
3. Making Sense of Systems
4. Planning in Pseudocode
5. Computational Thinking applies Everywhere

Computer Science Teacher Association Standards

Standard	Connection to Curriculum
1-AP-14: Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.	Students rearrange and modify the sequence in algorithms and in Try It activities to test and debug programs.
1B-AP-09: Create programs that use variables to store and modify data. Variables are used to store and modify data.	Students define and use variables in programs for the robot to store and retrieve data. For example, students create a variable for a sensor threshold value that is used in an operation compared to the sensor analog value.
1B-AP-10: Create programs that include sequences, events, loops, and conditionals.	Students write programs that include commands in a sequential order, repeated behaviors with loops, and decision-making behaviors with conditionals.
2-AP-13: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.	Students write programs decision-making behaviors by nesting if-else conditional statements inside loops to enable robot line tracking.
2-AP-19: Document programs in order to make them easier to follow, test, and debug.	Students practice pseudocoding and code commenting that allows for easier program comprehension, troubleshooting, and debugging.
2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.	Students make large programs simpler and organized by writing functions with parameters. For example, students recognize repeated behaviors in line tracking behaviors and turn those sections of code into functions so that those sections can be reused, and the program is easier to read.
2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.	Students control programs by using WaitUntil commands, conditional loops, and if-else statements. For example, to enable the robot to perform line tracking behaviors, students embed if-else statements that are nested within a conditional loop.
2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.	Students define and use variables in programs for the robot to store and retrieve data.
2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.	Students develop pseudocode to organize and sequence an algorithm that addresses a complex problem.
3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.	Students design programs to create solutions for model challenges that are based on real-world challenges robots face.

(Continued) Computer Science Teacher Association Standards

Standard	Connection to Curriculum
3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.	Challenges within the curriculum directs students to decompose problems into smaller components and build solutions up accordingly.
3B-DA-05: Use data analysis tools and techniques to identify patterns in data representing complex systems.	Sensors and other components on the robot collect analog values that the students analyze and apply to enhance the accuracy of their programs and better predict solution outcomes.
3B-AP-10: Use and adapt classic algorithms to solve computational problems.	Challenge activities result in the creation of a (simple) algorithmic solution and an accompanying program that implements it.
3A-AP-21: Evaluate and refine computational artifacts to make them more usable and accessible.	Students debug and modify their own programs to make improvements and compare the outcomes.

Computer Science Principles

Standard	Connection to Curriculum
LO 1.1.1: Use computing tools and techniques to create artifacts.	Challenge activities result in the creation of a (simple) algorithmic solution and an accompanying program that implements it.
LO 1.1.3: Analyze computational artifacts.	Students perform debugging on their own code, as well as debugging example solutions in practice programs and Check Your Understanding questions.
LO 1.3.1: Use programming as a creative tool.	Challenges are designed so that there are multiple solutions to a problem along with some extra challenges and extension activities. Students use programming to solve model challenges based on real-world events and can solve these challenges based on their own ability and agency.
LO 2.2.1: Develop an abstraction.	Robots gather information about the world through sensors, which turn physical qualities of the world into digital abstractions. Students must understand and work with this data to develop then implement their solution algorithms.
LO 2.3.1: Use models and simulations to raise and answer questions.	Students construct and use a “program flow” model of programming itself to understand how the robot uses data to make decisions and control the flow of its own commands.
LO 3.2.1: Use computing to facilitate exploration and the discovery of connections in information.	“Try It” activities allow students to uncover new concepts and make connections with the learning material and real-world applications of robotics.
LO 4.2.1: Express an algorithm in a language.	Students plan, write, and produce algorithms in VEXcode Pro.
LO 5.2.1: Use abstraction to manage complexity in programs.	Students use the process of abstraction by writing pseudocode and creating functions so that complex programs are organized, accurate, and easier to read.
LO 5.3.1: Evaluate a program for correctness.	Students perform debugging on their own code, as well as debugging example solutions in practice programs and Check Your Understanding questions.
LO 5.3.2: Develop a correct program.	Programmed solutions to challenges must work.
LO 5.4.1: Employ appropriate mathematical and logical concepts in programming.	Students use mathematical concepts in programming by writing conditional statements with comparison operators and finding relationships in the number of wheel rotations and distance travelled or degrees turned.
LO 7.2.1: Connect computing with innovations in other fields.	Students discuss how the process of abstraction and decomposition in programming and robotics can be applied to other subjects and fields of work.

Next Generation Science Standards

Standard	Connection to Curriculum
MS-ETS1-2: Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.	Solving challenges requires students to create and evaluate both hardware and software designs according to scenario scoring criteria.
MS-ETS1-4: Develop a model to generate data for iterative testing and modification of a proposed object, tool, or process such that an optimal design can be achieved.	When solving more difficult and complex challenges, students are guided toward iterative testing and refinement processes. Students must optimize program parameters and design.
HS-ETS1-2: Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.	Problem Solving methodology for challenges directs students to break down large problems into smaller solvable ones, and build solutions up accordingly; challenges give students opportunities to practice, each of which is based on a real-world robot
HS-ETS1-3. Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics as well as possible social, cultural, and environmental impacts.	Some Discussion Questions require students to consider real-world policies (e.g. requiring sensors on automobiles) based on the impact of that decision.

Common Core Mathematics

Standard	Connection to Curriculum
CCSS.MATH.CONTENT.7.G.B.4: Know the formulas for the area and circumference of a circle and use them to solve problems	Students calculate and use circumference to discover a proportional relationship with robot turns.
CCSS.MATH.CONTENT.7.RP.A.2: Recognize and represent proportional relationships between quantities.	Students use ratio language to describe and make use of the relationship between quantities such as Wheel Rotations and Distance Traveled.
CCSS.MATH.CONTENT.6.G.A.3: Draw polygons in the coordinate plane given coordinates for the vertices; use coordinates to find the length of a side joining points with the same first coordinate or the same second coordinate. Apply these techniques in the context of solving real-world and mathematical problems.	Students program shapes with specific (x, y, w, h) parameters to a coordinate system on the robot brain's screen.
CCSS.MATH.CONTENT.2.MD.A.1: Measure the length of an object by selecting and using appropriate tools such as rulers, yardsticks, meter sticks, and measuring tapes.	Students measure the distance the robot travels using a measuring tool.

Common Core English Language Arts

Standard	Connection to Curriculum
CCSS.ELA-LITERACY.RST.9-10.7: Translate quantitative or technical information expressed in words in a text into visual form (e.g., a table or chart) and translate information expressed visually or mathematically (e.g., in an equation) into words.	Students write pseudocode that consists of a hybrid language between written English and coding syntax and then translate it into code.
CCSS.ELA-LITERACY.RST.9-10.9: Compare and contrast findings presented in a text to those from other sources (including their own experiments), noting when the findings support or contradict previous explanations or accounts.	Students compare the results they find after the robot executes their program to how it is shown in the curriculum.

International Society for Technology in Education

Standard	Connection to Curriculum
1.5: Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions.	Students learn “Big Ideas” such as program flow, role of the programmer, planning in pseudocode, and more to develop problem solving skills in computer science.
1.5a: Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.	Students create solutions to challenges by using sensor data to predict outcomes, planning in pseudocode, and then translating the pseudocode into an algorithmic program which is a form of abstraction.
1.5b: Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem- solving and decision-making.	Students use sensors to collect data and program the robot to respond according to the data analyzed.
1.5c: Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.	During the process of pseudocoding, students decompose large solutions into smaller tasks in order to understand what exactly the robot is needed to do.
1.5d: Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.	Students write programs for the robot including sequences of commands and control commands so that the robot can complete challenges autonomously.

The Big Ideas

- Big Idea #1: Role of the Programmer

Students learn the roles and responsibilities of a programmer, which is to identify the task, plan out a solution, decompose that solution into steps that the robot can carry out, and then explain the steps to the robot through a program. As a programmer, it is important to keep programs precise. If you want the robot to do something, you need to communicate that idea with mathematical and logical precision, or it won't quite be what you intended. ¨

- Big Idea #2: Sense, Perceive, Plan, Act

Sensors provide information about the world. The program uses that information to figure out when to stop. And then the robot acts accordingly. Sense, Perceive, Plan, Act (SPPA) summarizes the four critical capabilities that every robot must have in order to operate effectively. ¨

- Big Idea #3: Make Sense of Systems

To understand the way something works, students construct a mental "model" of it that captures the important features and rules of the system. This helps with understanding Program Flow and how other similar systems work and execute programs. ¨

- Big Idea #4: Planning in Pseudocode

Students learn to plan complex programs in pseudocode so that their programs can be accurate and organized. The process of pseudocoding involves the decomposition of large tasks into smaller steps. And then abstracting those steps to be translated into code so that the robot can understand its task. ¨

- Big Idea #5: Computational Thinking Applies Everywhere

These skills – mathematical and logical clarity, using data, systems thinking with mental models, and problem solving – are not just for robotics. They are key to solving many problems in the world.