Can Computational Thinking Practices Be Taught in Robotics Classrooms?

Presented at the International Technology and Engineering Education Conference

National Harbor

Washington DC

March 2-4, 2016

Authors

Robin Shoop, Jesse Flot, Tim Friez **Carnegie Mellon University, Robotics Academy**

Christian Schunn, Eben Witherspoon **University of Pittsburgh, Learning Research and Development Center**

This material is based on work supported by the National Science Foundation under DRK-12 research, Award Number 1418199, *Changing Culture in Robotics Classrooms. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or any collaborator or partner named herein.*

Why Teach Computer Science and Computational Thinking in Robotics Class?

Robotics competitions have inspired over thirty-five thousand robotics teams in the United States (FIRST, RECF, BEST, 2014) and are the driving force for schools adopting robotics classes. However, in terms of teaching computer science, robotics competition challenges typically emphasize low-level control problems such as driving in a straight line without drifting to the side, finding an IR beacon and driving towards it, or picking up, stacking, or throwing an object. Even fully autonomous competitions at the middle and high school levels skew toward static sequences of commands with robots that execute tasks in order. Many of today's robotics classrooms replicate activities found in robotics competitions; they emphasize mechanical design and construction and teach only basic level behavior-based programming. This stands in stark contrast with innovation in modern robot systems such as machine vision and learning (Guizzo & Deyle 2012), multi-sensor robot localization (Velosa, 2013), and multi-agent communications (Matignon, 2012; Robotics Virtual Organization 2013). This paper reports on the progress made on the National Science Foundation DRK-12 research, Award Number 1418199, *Changing Culture in Robotics Classrooms*. This research project's goal is to build a scaffolded set of tools that can be used in robotics classrooms that teach the big ideas found in the Computer Science Principles (CSP) (Astrachan, et al., 2013) and engage students in activities where they apply Computational Thinking Practices (CTP) (Bienkowski, et al., 2015) as they problem solve.

The partners on the project include: Carnegie Mellon's Robotics Academy (CMRA), the University of Pittsburgh's Learning Research and Development Center (LRDC), and Robomatter Incorporated (Robomatter). In this publication CMRA, LRDC, and Robomatter are referred to as "the Team". The project builds on CMRA's broad reach into the robotics education community and LRDC's years of experience with the construction and evaluation of Design Based Learning units. Robomatter is a Carnegie Mellon inspired company whose mission is to develop Computer Science and STEM solutions for education. Competition and dissemination partners include the Robotics Education and Competition Foundation (RECF) and Innovation First VEX Robotics (VEX). RECF and VEX promote these activities to over 12,000 teams that participate in RECF robotics competitions annually.

Can Computational Thinking Practices Be Taught in Robotics Classrooms?

Computational Thinking has been defined as "a thought process that involves formulating problems and solutions so that the solutions are represented in a form that can be effectively carried out by an information processing agent" (Cuny, Snyder, Wing, 2013). There is a growing recognition that computer science and computational thinking are new basic skills that all K-12 students must learn (SEA 2015, ESSA 2015, McKinsey, 2013, College Board, 2016). Robotics provides opportunities to integrate and teach programming (Touretzky, 2012, Liu, A. et. al., 2013), engineering design (VEX, 2014, Dye et. al., 2005), and mathematics (Schunn et. al., 2007, 2010, 2011), all areas that benefit from computational thinking (Wing, 2006).

In addition to teaching key content knowledge in these areas, the Team's tools and strategies reinforce Computational Thinking Practices (Beinkowski, et. al., 2015, College Board, 2016) such as: connected computing, creating computational artifacts, analyzing problems and artifacts, and communicating and collaborating when solving computational projects. Changing Culture in Robotics Classrooms activities place students in designed learning environments where they are required to solve problems using CTP like abstraction, decomposition, and algorithms. The Team believes that designed robotics problems that require students to apply the Big Ideas found in CSP and CTP will better integrate their understanding of core computer science concepts with an ability to use that knowledge to solve meaningful problems.

Strategy Overview

Our project promotes computational thinking and programming by augmenting existing robotics classrooms and competitions with:

- Curriculum that supports beginning, intermediate, and advanced robot programming grades 5-12.
- Mapped badged pathways within the curriculum that lead to Computer Science (CS) related certifications.
- Robot Virtual World (RVW) programmable activities used with the curriculum and designed to scaffold student access to higher-order CSP related challenges.
- Modified Autonomous Competitions (MACs) that are similar to the annual robot competitions that thousands of schools compete in, but are optimized to enable teachers to foreground CTP.
- Teacher training that integrates CSP and CTP into robotics teacher professional development classes.

Integrating CSP into Robotics Classrooms

Many middle and high school robotics courses are established as a direct result of the school's participation in a robotics competition. The classes themselves are frequently used to prepare for the competition. Teachers typically provide students with various forms of existing robotics curricular units and practice competition activities with varying degrees of focus on programming skills. It is therefore highly likely that students develop some basic programming skills (e.g., basic movement control commands and simple sensor-triggered behaviors), but there is little reason to believe that they proceed to any higher levels, especially given that the competition challenges generally neither require nor reward more sophisticated programming. Many robotics competitions start in the fall (FTC, VEX, BEST, FLL) and consist of a six-to-eight-week intensive engineering experience. A yearlong (i.e., 36 week) course is left with 28-30 weeks of class time to cover additional topics. Our plan builds on this established competition cycle and uses the time after the competition to engage students in deeper CSP and CTP related lessons and activities (See Table 1).

	Table 1. Proposed Robotics Course Schedule		
	Fall	**Winter**	**Spring**
Activity	Robotics Competition	Modified Autonomous Competition (MAC)	CS Principles Robotics Unit
What students do	Students compete in an existing robotics competition	Students compete in a virtual competition designed to stress sophisticated autonomous operation	Students engage in design-based learning units in which they design algorithmic solutions to robotics problems
Primary content	Mechanics and Basic Programming, ideation and creativity	Autonomous Programming with an emphasis on decision-making logic	High-level problem solving, abstraction, and algorithmic design
Relevant CS Principles Focus Areas	Big Idea 1 Creativity Big Idea 5: Programming (Basic)	Big Idea 1: Creativity Big Idea 2: Abstractions Big Idea 5: Programming	Big Idea 1: Creativity Big Idea 2: Abstractions Big Idea 4: Algorithms
Robot Virtual World (RVW) Role	Direct simulation of robotics challenge to support teams in competition	Primary activity platform: MAC & RVW competitions and activities	Higher order problems are posed and solved in a RVW environment

Key Technologies: MACs and RVWs

Modified Autonomous Competitions (MACs) look similar to their real-world counterparts and can be programmed using multiple programming languages that are popular in education. MACs enable students to compete in robotic competitions that are designed to foreground and reward programmable solutions as opposed to mechanical solutions. For example, MACs are designed to simplify mechanically complex tasks like grabbing, stacking, and hanging by utilizing video game-like mechanics that simplify the physics in the virtual simulations (RECF 2012, 2013, 2014). Mechanically challenging tasks can lead to confusion and frustration when learning to program. MACs abstract away these mechanically difficult tasks, enabling students to focus on the development of algorithms and computational artifacts instead of engaging in mundane exercises involving robot positioning.

Robot Virtual Worlds (RVWs) can be used to provide students with advanced sensing systems that are commonly used in robotics today (Velosa, 2013) but not available in most educational settings, like a Virtual Global Positioning System (V-GPS). These advanced sensing capabilities enable the Team to teach deeper computer science concepts in the advanced level course. In addition, earlier studies (Flot et al., 2012, 2013, Liu et. al., 2013) found students completed learning activities in many fewer weeks when using the RVW technology than when teachers were attempting to complete the same basic activities and teach the same concepts using a physical robot. Finally, teaching students to program in simulated environments reflects current workplace practice (Herbert, et al, 2011).

RVW/ MAC History

Since 2012, CMRA has developed RVW simulations of middle and high school Robotics Education and Competition Foundation (RECF) robotics competitions (CS2N, 2012, 2013, 2014, 2015). In 2015, RECF and Robomatter developed a virtual competition for teams that includes a $5,000 scholarship, opportunities to earn a certification, and an invitation to the RECF sponsored VEX Robotics World Championship. The RVW simulations are designed to provide students with a programmable sandbox where they can develop strategies and code that they can eventually load onto and test with their physical robot; RVWs also provide each student with their own individual virtual robot to program and experiment with. Over the years, the team has found that some RECF games make better MACs than others.

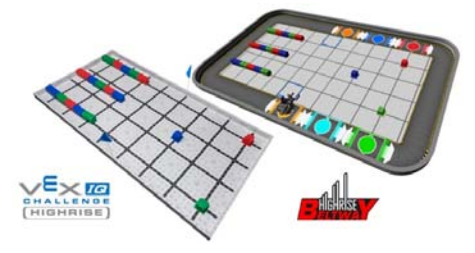


Figure 1 – The RECF Highrise game (left) and Beltway MAC game (right).

In 2014, the RECF developed the VEX IQ Challenge, Highrise. Highrise is played on a grid-like field and involves moving and stacking 3 inch colored blocks (Figure 1). RECF and VEX Robotics made the Highrise game an "evergreen" game; all game elements will be available for the foreseeable future for classroom use. The Team evaluated the Highrise Challenge and found multiple opportunities to design a MAC to promote the concepts of algorithms, decomposition, abstraction, creativity, and programming—all Big Ideas found in CSP. Therefore, the Highrise competition was selected as a model for the design of a new MAC, Beltway. In the next section, we discuss they key properties of the Beltway game in supporting student learning.

The Beltway Game & Its Educational Properties

The Beltway game (figure 2) is played on a grid-like field that enables students to develop algorithmic functionalized solutions for robot path planning; the distance the robot travels and the angle that the robot turns. The virtual robot includes a color sensor that can be used to count lines and detect colored blocks, a gyroscope sensor to control turns, an ultrasonic sensor that senses distance away from objects, smart motors with built-in motor encoders and Touch LED sensors that are used for debugging and to program the belt (unique to the MAC). The Beltway game involves moving and stacking the twelve green, twelve red, and twelve blue blocks — the blocks on the left side of the table are scored by stacking them in tower formations onto the fixed blocks located on the right side of the table. Players receive points for moving any color block to the right side of the table, but they receive a multiplier effect if they stack the matching colored blocks on top of fixed blocks. For

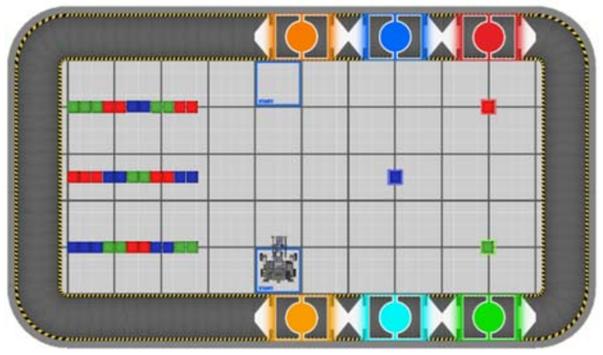


Figure 2 - Top down view of the Beltway game.

example, if a red block from the left side of the table is stacked onto the red block on the right side of the table that stack of blocks receives a multiplier to their base point value. Unique to the Beltway game is an automated track with programmable stopping points.

An Engineered Solution Designed to Solve the Accumulated Error Problem

One of the larger challenges in educational robotics involves a phenomena called “accumulated error”. Accumulated error occurs when a robot’s movements are off by a small amount from the expected behavior. If each movement is off by a small amount, then each subsequent movement magnifies the error and introduces variability to the robot trying to accomplish a more complex task. For example, if a robot is programmed to turn 90 degrees and then to move forward 1 meter, but actually turns only 89 or 91 degrees and then moves forward, the further that the robot travels in the forward direction, the further the robot’s actual location will deviate from its desired location. A common strategy that robot programmers use to solve the incremental error problem is to reset the location of the robot when the robot gets to a known position in the world. Beltway provides game players with an easy-to-access known position in the world via a programmable track that surrounds the grid world. The programmable track is also known as the “Beltway” (figure 2). The programmable track enables programmers to relocate the robot’s position in the following ways:

1. When a robot drives onto the beltway, regardless of the angle the robot approaches the beltway, the robot’s body is automatically repositioned so that the front of the robot is perpendicular to the beltway.
2. Once the robot enters the track, the track can be programmed to carry the robot in a clockwise or counterclockwise direction to a fixed, known location.
3. The known locations are six color detectable locations on the track: light orange, light blue, green, red, dark blue, and dark orange. The track spins, but the color coded locations stay in the same position. The programmer sets the programmable touch LED (an additional sensor) to display the corresponding color to indicate to the track which location it should stop.

Simplifying Mechanically Difficult Challenges

A student can spend an inordinate amount of time developing a program that consistently picks up a block, only to have the physical robot fail to pick up the block because their robot drifted off target (due to accumulated error) or the block had been bumped. In order to make the game more solvable using mathematically correct algorithmic programmable solutions, the Team made a conscious effort to reduce the complexity of the mechanically challenging tasks (such as picking up and stacking the blocks) using video “game mechanics” which modify the physics within the game.

Simplifying Picking Up Blocks – In order to make it easier to grab and then lift the blocks, the team incorporated a magnetic like zone into the block-facing portion of the gripper (see Figures

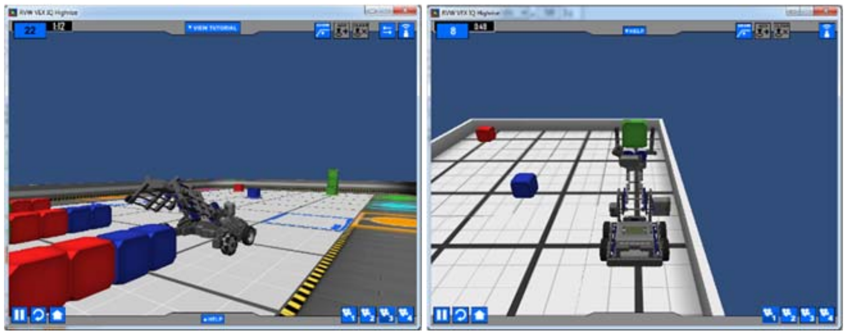


Figure 3 on the left shows a robot ready to pick up a block in the Beltway game. Figure 4 on the right shows a robot carrying the block in a simulated version of the actual Highrise game.

3 & 4). If a block is within the magnetic like zone, then the robot gripper automatically sucks the block into the gripper when signaled to close. When the robot's gripper is programmed to open, the magnetic zone is turned off and the block is released as expected.

Simplifying Stacking Blocks – Stacking blocks in the real world requires precision and the blocks often fall after they are stacked. The Team added a game mechanic to the world so that when a robot is holding a block in its gripper and is positioned to stack the block onto an existing tower, the tower automatically adjusts to allow the stack to remain standing (see Figure 5). This playful element enables short robots to stack blocks many times higher than the robot appears to be able to reach (see Figure 6). This game mechanic causes all of the blocks to pop into the air and then settle back onto the new stack of blocks. The team designed this game mechanic to prevent stacks from tipping over. In sum, the player is able to continue to add blocks regardless of the height of the stack.

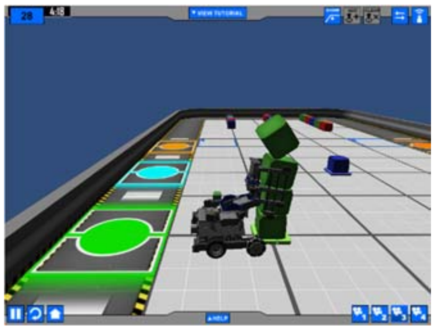


Figure 5 shows a robot stacking a block

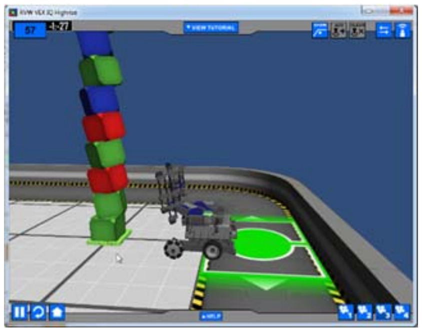


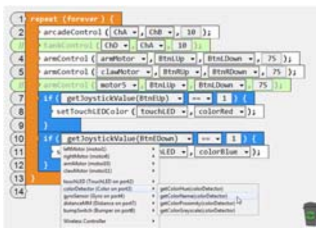
Figure 6 shows eight blocks stacked onto of each other. Even though there is an allusion that they might fall, the game physics prevented them from falling.

Beltway Summary

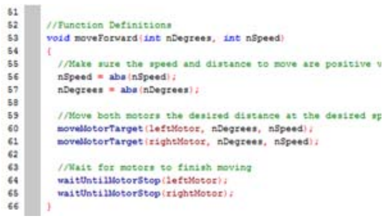
The Beltway MAC takes advantage of: 1) the Highrise grid world, 2) a programmable belt designed to solve the accumulated error problem, and 3) specially designed game mechanics that alter the physics in the game to design a game that can be solved using programmable algorithms.

Methods of Scaffolding Instruction

Students can program in the game using either ROBOTC Graphical (RBG) or ROBOTC text-based (ROBOTC) programming languages. The RBG programming language is designed for younger students as well as new programmers to enable them to quickly develop working code in terms of robot behaviors without being overloaded by the additional complexities of remembering command names and obscure language-specific syntax (see Figure 7 and 8). For example, RBG enables students to use a prebuilt function command called *forward* to program their robot to move forward for a specific distance. The *forward* command has default settings designed to move a robot in the forward direction for one wheel revolution at 50% power. *forward* also includes drop down menus that enables the student to control the distance traveled via the following easy to program parameters: wheel rotations, wheel rotation angles (in degrees), or timing (milliseconds, seconds). *forward* also includes an editable text box called ‘speed’ that enables students to control the power level of their robot. In addition, RBG includes other basic movement commands: *backward*, *turnLeft*, *turnRight*, and *moveMotor* which utilize the same parameters. These function-like commands build student confidence by enabling programmers to have immediate success programming their robots.



```
repeat (forever) {
  arcadeControl ( ChA , ChB , 10 );
  tankControl ( ChD , ChA , 10 );
  armControl ( armMotor , BtnLUp , BtnLDown , 75 );
  armControl ( clawMotor , BtnRUp , BtnRDown , 75 );
  armControl ( motor5 , BtnLUp , BtnLDown , 75 );
  if ( getJoystickValue(BtnEUp) == 1 ) {
    setTouchLEDColor ( touchLED , colorRed );
  }
  if ( getJoystickValue(BtnEDown) == 1 ) {
    s LED , colorBlue );
  }
}
```

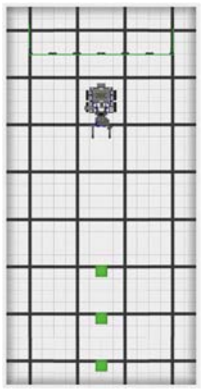


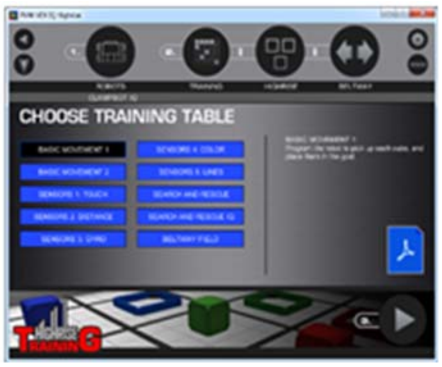
```
//Function Definitions
void moveForward(int nDegrees, int nSpeed)
{
  //Make sure the speed and distance to move are positive v
  nSpeed = abs(nSpeed);
  nDegrees = abs(nDegrees);

  //Move both motors the desired distance at the desired sp
  moveMotorTarget(leftMotor, nDegrees, nSpeed);
  moveMotorTarget(rightMotor, nDegrees, nSpeed);

  //Wait for motors to finish moving
  waitUntilMotorStop(leftMotor);
  waitUntilMotorStop(rightMotor);
}
```

Figure 7 on the left shows ROBOTC Graphical, a drag and drop programming language designed for younger children. Figure 8 on the right shows sample ROBOTC code.

Full ROBOTC uses a text-based, ANSI-C compliant version of the C programming language which has been optimized to include robot-specific commands and features. Both versions of ROBOTC (Text and Graphical) are fully supported with robotics curriculum which is posted at the Carnegie Mellon Robotics Academy website.

The Team uses an Understanding by Design (Wiggins & McTighe, 2005) approach to curricular development. The Team evaluated the skills needed to successfully solve the Highrise game and developed a scaffolded set of training tools that require students to apply CSP and CTP (see





Figures 9, 10, 11 - The first example challenge (left) involves programming a robot to move forward, pick up three cubes and deliver them to a specified area on the table. Each successive challenge teaches a new programming concept or how to use a different sensor. This initial challenge is able to be solved using values from the smart motor encoders, a sonar sensor, a gyro sensor, or a combination of all three sensors. The Robot Programming Chapter/Certification breakdown (Center) shows the badges and mapped badged pathway the team developed. The Beltway Training Table interface (Right) shows the user interface for the challenges inside of the Robot Virtual Worlds software.

Figure 9, 10, and 11). The curriculum includes: step by step instruction and extension activities that foreground CSP and require students to apply CTP, RVW simulated challenges that provide each student with their own programmable robot, and a mapped badged pathway (Abramovich, et. al, 2011, 2013) that leads to an "Introduction to Robot Programming Certification" (CS2N, 2012). These same skills are useful to solve the Beltway Challenge as well.

Foregrounding CSP and CTP in our New Curriculum

It is important to note that robotics education is a relatively new subject area in which both teachers and researchers are still developing best teaching practices. Robotics integrates computer science and STEM concepts, systems, and technologies and it can be challenging to determine what to foreground in each lesson. In earlier research (Schunn, et. al. 2008) the Team learned that in order to see educational gain in the development of a skill or concept in robotics classes that the skill or concept needs to be foregrounded in the robotics lesson. By foreground we mean: tell students what skill or concept that the lesson or activity teaches, share the evaluation rubric that will be used to assess the lesson, engage the student in the lesson or activity, and then evaluate their performance. To facilitate this, each lesson includes teacher notes that explain which CSP is foregrounded in the lesson, what CTP will be used, how the CTP is assessed, the instructional aids that are included with the lesson (i.e. rubrics, videos, MAC and/or RVW), and a scope and sequence that describes how the lesson is implemented and how much time that the lesson will take.

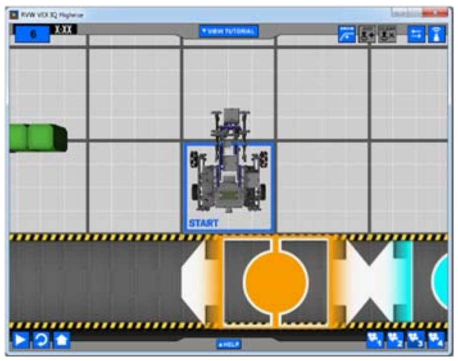


Figure 12 - Students are challenged to develop pseudocode to pick up the green blocks at the left of the start position. This activity enables the teacher to talk about programming in terms of decomposition and creativity.

For example, creativity is a big idea in CSP - Directly below is an example of how a teacher might foreground the concept of creativity as students begin to solve Beltway.

Assign students to work in pairs and develop algorithmic strategies to move from the start position to pick up the green block, see Figure 12. After students have brainstormed solutions have them present their favorite solution to the class.

Below are examples of behaviors and algorithms that students might suggest.

- Point turn to the left, open the gripper, drive to the green block, close the gripper, lift the gripper arm.
- Move forward to the black line on the grid, turn 90 degrees left using the gyro sensor, raise the gripper arm, open the gripper, track a line until the sonar sensor detects the block, close the gripper, raise the gripper arm.
- Turn left until the value of the motor's encoder is greater than the target encoder value, raise the gripper arm for 90 encoder counts, open the gripper, lower the arm, and close the gripper.

The next level of the lesson is to determine the most efficient solution that can be turned into a reusable function. At each part of the lesson students are required to analyze and defend their solution. This foregrounds the CTP concept of communicating and collaborating when solving computational projects. All lessons require students to work in teams, develop prototype computational artifacts, present and defend their solutions in presentations, and then refine their solution to the problem. This example demonstrates one way a teacher might utilize the tools and strategies embedded in our curriculum to build conceptual understanding of a key programming concepts (i.e. algorithm development), while integrating this knowledge with Computational Thinking Practices.

Intermediate Level CSP and CTP Activities

The Intermediate Level (IL) curriculum assumes that students have a basic understanding of the physical robot's characteristics, robot mathematics, algorithms, decomposition and program flow, and conditional statements; these concepts were covered in the Beginner Level Curriculum. The IL curriculum emphasizes the use of variables, parameters, functions, and arrays. Students work in teams providing opportunities for the instructor to emphasize the benefit and challenges of communications and collaboration.

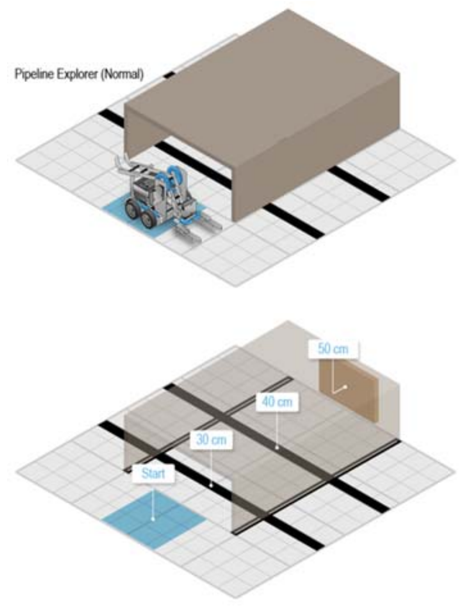


Figure 13 The robot starts at the blue square and moves through the tunnel until it touches a wall. After returning, the robot displays the locations of the two lines on its LCD screen.

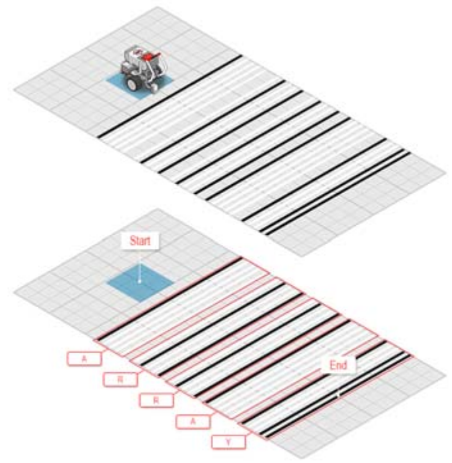
Intermediate level challenges requires students to develop systems that collect, store, and use data. Several example challenges are: The Pipeline Explorer, the Barcode Reader, and the Automated Garage Storage System.

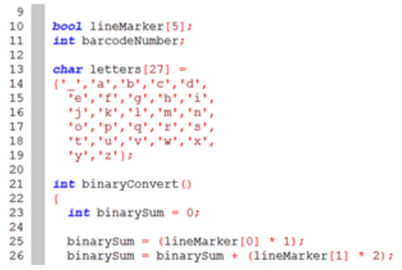
The Pipeline Explorer

The Pipeline Explorer module is divided into six units: Encoders and Variables, Go One Line Function, Move Multiple Lines, Convert Distance to CM, Passing Parameters, and Displaying Values to the LCD. Each of those units is further divided into smaller lessons which help to teach the major concepts of the chapter. Students are required to build prototypes (Figure 13) and debug mini programming challenges presented in each chapter. At the conclusion of the module, there is a capstone challenge that requires the students to present their working prototypes and explain their code.

The Bar Code Scanner

In this lesson, students develop a working prototype of a barcode scanner. In the barcode scanner module, the robot's color sensor will read the "data" and the students program will convert it into a message. The challenge requires the robot to scan multiple sets of barcodes





```

bool lineMarker[5];
int barcodeNumber;

char letters[27] =
{'_','a','b','c','d',
  'e','f','g','h','i',
  'j','k','l','m','n',
  'o','p','q','r','s',
  't','u','v','w','x',
  'y','z'};

int binaryConvert()
{
  int binarySum = 0;

  binarySum = (lineMarker[0] * 1);
  binarySum = binarySum + (lineMarker[1] * 2);
```

Figure 14 the robot reads the barcode scanner and student developed code interprets the message.

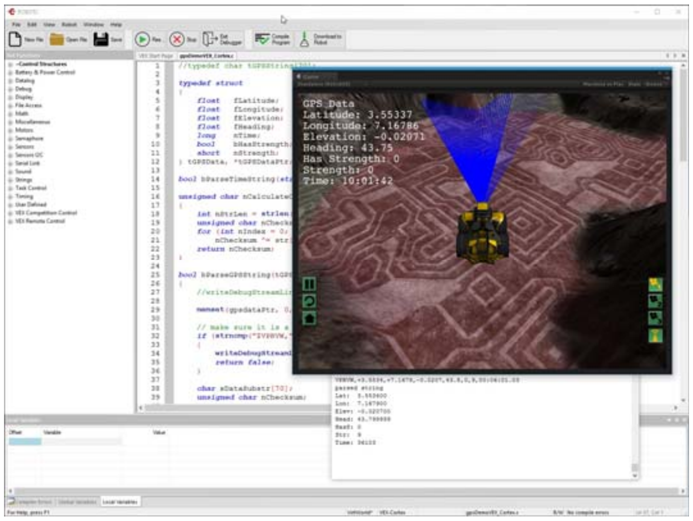
(Figure 14) in order to convert the black and white segments into a message. In this lesson students are required to learn: the binary numbering system, how to program arrays, how to identify an element within an array, how to use Boolean variables when programming, how to utilize a for-loop to efficiently store data in an array, how to use the counter variable that is part of the "for" loop to iterate through and store information sequentially in an array, how to convert the array of Boolean values into binary numbers, how to convert the Boolean number algorithm into alphanumeric characters, and how to display values onto the LCD screen.

The Automated Garage Storage System

This problem begins by presenting student teams with a request for proposal from a fictitious customer asking for the following: Develop a scale model working prototype of an Automated Garage Storage System that parks miniature cars. The prototype must be structurally sound and contain at least four parking spaces. The system must be able to identify cars and parking spaces using a color-coded system. This will require the use of at least two sensing systems (or input devices). An LCD screen will be used for communications between the automated system and the driver. The LCD screen will ask the driver to indicate whether the car is entering or the car is exiting. The driver needs to be able to scan a colored card to identify which parking space the driver wants to access. The robotic system should then move the car to the correct available space. While moving, an alert will sound to warn others of the automated parking system's movements. The miniature cars can be manually picked up and placed in spaces as needed during the prototype demonstration. This culminating challenge provides an iterative research and design problem that includes open-ended problem solving and opportunities for students to engage in CTPs and apply intermediate level programming concepts.

Current Development Efforts – Advanced Level Curriculum

The Team's goal in year three of the project is to develop an Advanced Level Curriculum for high school level CS Robotics Classes. The new course required the team to develop new technology that provides students access to the types of sensing systems being used in today's real-world robotic systems. Figure 15 shows that the Team has designed and integrated a Virtual Global Positioning System (V-GPS) system into the RVW simulation environment. The V-GPS system provides student access to the Longitude, Latitude, Elevation, Heading, and Speed of the robot, along with additional world parameters including Satellite Strength and World



```
//typedef char tGPSString[70];

typedef struct
{
    float   fLatitude;
    float   fLongitude;
    float   fElevation;
    float   fHeading;
    long    nTime;
    bool    bHasStrength;
    short   nStrength;
} tGPSData, *tGPSDataPtr;

bool bParseTimeString(st

unsigned char nCalculate

{
    int nStrLen = strlen
    unsigned char nChecks
    for (int nIndex = 0;
        nChecksum ^= str
    return nChecksum;
}

bool bParseGPSString(tGPS
{
    //writeDebugStreamLi

    memset(gpsdataPtr, 0,

    // make sure it is a
    if (strncmp("$VRVW,"
    {
        writeDebugStream
        return false;
    }

    char sDataSubstr[70];
    unsigned char nChecksum;
```

```
GPS Data
Latitude: 3.55337
Longitude: 7.16786
Elevation: -0.02071
Heading: 43.75
Has Strength: 0
Strength: 0
Time: 10:01:42
```

```
parsed string
Lat:  3.553600
Lon:  7.167900
Elev: -0.020700
Head: 43.750000
HasS: 0
Str:  0
Time: 36102
```

Figure 15 shows how the team integrated a programmable Virtual GPS sensor into the RVW system.

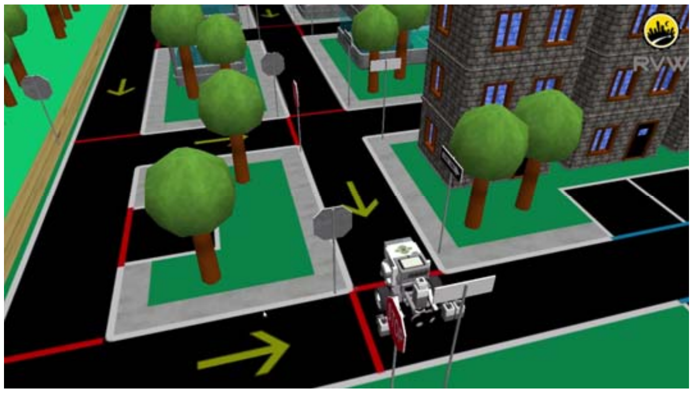


Figure 16 shows a version of a mini Urban Challenge that is available today for students to practice advanced programming concepts with.

Time. With this data, rather than relying on a linear set of commands and a static world for robot locomotion, students can program their robots to move to a specific location within the virtual world, even if their starting position changes or the robot gets off course.

The team will also introduce variability into the RVW system (i.e. obstacles and conditions that randomly appear). Typical robot challenges found in middle and high school classrooms have the robot travel from point A to point B. The RVW simulation environment enables the Team to pipe data into the world that automatically changes the problem and requires the use of algorithmic solutions. The combination of V-GPS sensor data with information from sensors such as an Ultrasonic Sensor enable students to program their robots to move to a specific location when obstacles or other variability is introduced into the world.

By creating learning environments that foreground computational thinking practices, the team hopes to prepare students for future innovation.

In Conclusion

Computational thinking is a new basic skill that all students benefit from. This project is only in its second year and has made significant progress developing instructional tools and technology designed to facilitate learning. However, the Team is currently looking for additional school based partners who are committed to testing these new innovations in middle of high school settings. Current school based partners have been successful in implementing portions of the curriculum, but few school districts have yet to see what the benefit from full implementation of the complete curriculum sequence might yield. We are seeking serious partners.

To learn more about the project go to the Robotics Academy website: www.eduction.ri.cmu.edu

To learn more about Robot Virtual Worlds go to: www.robotvirtualworlds.com

To find virtual competitions go to the Computer Science STEM Network: www.cs2n.org and select the competition tab.

References

Abramovich, S., Higashi, R., Hunkele, T., Schunn, C., & Shoop, R. (2011). *An achievement system to increase achievement motivation*. Paper presented at the Games Learning Society 7.0, Madison, WI.

Abramovich, S., Schunn, C., & Higashi, R. M. (2013b). *Are badges useful in education?: it depends upon the type of badge and expertise of learner*. Educational Technology Research and Development, 1-16.

Astrachan, O., Briggs, A., Diaz, L., (2009-13) *CS Principles*, http://www.csprinciples.org/ NSF Special Projects O938336

BEST Robotics Annual Report – 2012 Report states that they had over 800 teams all from schools
http://www.roboticseducation.org/wp-content/uploads/2013/05/RECF_AnnualReport2013.pdf

Brennan, K., Resnick, M., (2012), *New Frameworks for Studying and Assessing the Development of Computational Thinking*, Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada

Bienkowski, M., Snow, E., Rutstein, D. W., & Grover, S. (2015). *Assessment design patterns for computational thinking practices in secondary computer science: A first look* (SRI technical report). Menlo Park, CA: SRI International. Retrieved from http://pact.sri.com/resources.html

College Board, The (2013). Computer Science: Principles; Big Ideas and Key Principles, Learning Objectives and Evidence Statements.

College Board, The (2016) *AP Computer Science Principles Curriculum Framework* 2016-2017. https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-curriculum-framework.pdf

CS2N, 2012, 2013, 2014, 2015 – *The Computer Science Student Network*, Find current and past virtual robotics competitions http://www.cs2n.org/competitions

Cuny, J., Snyder, L., Wing, J., (2010), *Demystifying computational thinking for non-computer scientists* Unpublished manuscript in progress, referenced in http://www.cs.cmu.edu/~ CompThink/resources/TheLinkWing. pdf

Dym C. K., Agogino A., Eris O., Frey D., Leifer L., *Engineering Design Thinking, Teaching, and Learning*, Journal of Engineering Education 94 (1), 103-120

ESSA, 2015 *Every Student Succeeds Act*, Public Law No: 114:95, Passed 114th Congress 12/10/2015

FIRST. (2013). http://www.usfirst.org/aboutus/first-at-a-glance reports that they have 32,650 teams with 350,000 students participating and over 120,000 adult volunteers.

Flot, J., Schunn, C., Lui, A., Shoop, R. (2012). *Learning how to program via robot simulation*. Robot Magazine, 37, 68-70.

Guizzo, E., Deyle, T., (2012), *Robotics Trends for 2012*, IEEE Robotics and Automation Magazine, March 2012

Herbert, M., Thorpe, C., Stenz, T., (2011), *Intelligent Unmanned Ground Vehicle Autonomous Navigation* Research at Carnegie Mellon. Springer Science + Business Media. LLC

Liu, A., Schunn, C. D., Flot, J., & Shoop, R. (2013c). *The role of physicality in rich programming environments*. Computer Science Education, 23(4), 315-331. 10.1080/08993408.2013.847165

Liu, A., Newsome, J., Schunn, C. D., & Shoop, R. (2013d). *Kids learning to program in about half the time*. Tech Directions, March, 16-19.

Matignon, L, Jenanpierre, L, Mouaddib, A, (2012), *Coordinated Multi-Robot Exploration Under Communication Constraints Using Decentralized Markov Decision Making Processes*. Proceedings of the Twenty Sixth AAAI Conference on Artificial Intelligence, 2012

McKinsey 2013, *Disruptive Technologies That Will Transform Life, Business, and the Global Economy*. McKinsey & Company 2013

Robotics Virtual Organization. (2013), *A Roadmap for U.S. Robotics—From Internet to Robotics* (2013).
Retrieved from http://robotics-vo.us/sites/default/files/2013%20Robotics%20Roadmap-rs.pdf

Schunn, C.D., & Silk, E. M. (2011). *Learning theories for engineering technology and engineering education*. In M. Barak and M. Hacker (Eds.), Fostering Human Development through Engineering and Technology Education (p. 3–18). Sense Publishers.

Schunn, C. D., Silk, E. M., Higashi, R., & Shoop, R. (2010). *Designing technology activities that teach mathematics*. The Technology Teacher, 69 (4), 21-27.

Schunn, C. D., & Silk, E. M. (2008, June). *Using robotics to teach mathematics: Analysis of a curriculum designed and implemented.* Paper presented at the annual meeting of the American Society for Engineering Education.

Schunn, C. D., & Silk, E. M. (2011, April). *Resources for learning robots: Facilitating the incorporation of mathematical models in students' engineering design strategies*. Paper to be presented at the Annual Meeting of the American Educational Research Association, New Orleans, LA, USA.

Schunn, C. D., Silk, E. M., & Shoop, R. (2009). Synchronized robot dancing: Motivating efficiency & meaning in problem-solving with robotics. *Robot Magazine*, 17 , 74-77.

Schunn, C. D., Silk, E. M., Higashi, R., Shoop, R., Dietrich, A., & Reed, R. (2007). The use of robotics to teach mathematics. *Robotics Educators Conference*, Butler, PA, USA.

SEA 2015, STEM Education Act 2015, House Bill 113HF031, http://docs.house.gov/billsthisweek/20150223/SMITTX_1020_xml.pdf

Touretzky, D., (2012), *Seven Big Ideas in Robotics, and How to Teach Them*, SIGCSE Conference, March, 2012 ACM 987-1-4503-109807

Bienkowki, M., Snow, E., Rutstein, D.W., Grover, S., (2015), *Assessment Design Patterns for Computational Thinking Practices in Secondary Computer Science: A First Look* (SRI Technical Report). Menlo Park, CA: SRI International. Retrieved from http://pact.sri.com/resources.htm/

Veloso M., Biswas J., (2013), *Multi-Sensor Robot Localization for Diverse Environments*, RoboCup Symposium, 2013

VEX Robotics 2014 Annual Report -
http://www.roboticseducation.org/documents/2014/05/recf-annual-report.pdf
Report states that there are 10,107 teams.

Wiggins, G., McTighe, J. (2005) *Understanding by Design*. Pearson Education, Upper Saddle River, NJ, expanded 2nd edition, 2005.

Wing, Jeannette M. (2006) "*Computational Thinking*", CACM Viewpoint, March (2006): 33-35.