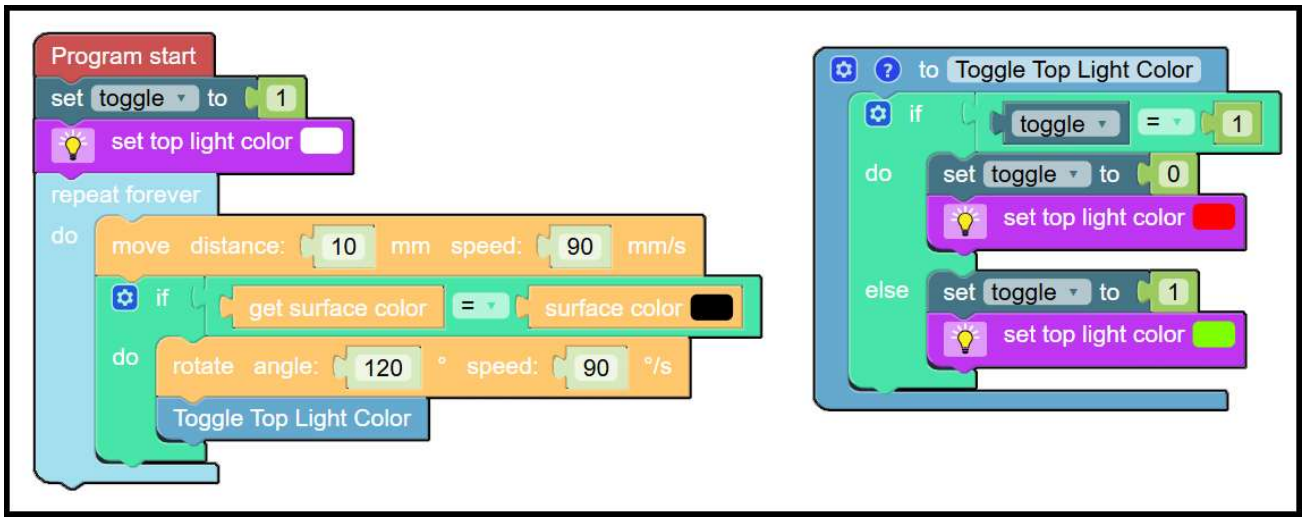
OzoBlockly Editor—**Keep Ozobot in Jail Challenge**—Student Guide

Grades 6 - 9

by Richard Born, Associate Professor Emeritus, Northern Illinois University

If you haven't already done so, you should read the *Keep Ozobot in Jail Student Guide*. Doing so will familiarize you with the challenge details expected from the student. The author's OzoBlockly Editor solution is shown in the figure below. The share code for this solution is **4hfvvvu**.



The solution consists of the main program on the left and one subroutine function *Toggle Top Light Color* on the right.

The program starts with the value of the variable *toggle* set to 1 and the top light color set to white. Following that is a forever loop in which Ozobot moves 10 mm, about half the width of the heavy black jail border. If the surface color is black, meaning that Ozobot has encountered a border, it then rotates 120 degrees and toggles the color of the top light. If the surface color is white, meaning that Ozobot is still within the confines of the jail, then it repeats the loop and again moves 10 mm. This loop continues indefinitely until *Stop* is selected on the editor screen or Ozobot's battery runs out of charge.

The *Toggle Top Light Color* subroutine checks to see if the curent value of the variable *toggle* is 1. If so, it changes *toggle* to 0 and sets the top light color to red. Otherwise, it changes *toggle* to 1 and set the top light color to green.

It will take a bit of experimentation for the student to come up with an angle of rotation that works well. As stated in the Editor references, Ozobot performs rotations with a limited precision (*e.g.* Ozobot may turn about 85 to 93 degrees instead of 90). This is due to traction differences between different operating surfaces, speed calibration, and its hardware design limitations.

You can see that the solution to the *Keep Ozobot in Jail Challenge* involves the use of several computer science programming concepts:

- The use of variables to hold the values of program constants
- Subroutines to break up code into logical segments
- The use of loops to handle tasks that must be repeated several times
- The use of conditionals, as needed, to control the flow of logic
- Subroutine and variable names that support self-documentation
- Nesting of structures

Ways to Use This Challenge with Your Students

1. At the simplest level you can present the author's solution to the students and help them do a code review like that presented above.
2. You could present the challenge to the students specifying that they must make use of the six programming concepts discussed in the previous section of this document.
3. You can present the challenge to the class without any indication of how to code it in the OzoBlockly Editor. Students then come up with their own solution, based on the challenge requirements in the Student Guide. Students could present their solutions to the class for a discussion of the advantages and disadvantages. Students may even come up with a "better" solution than the author's solution!

Regardless of the approach that you choose, it is suggested that students view the accompanying video that shows Ari in motion in the jail. This will give them a chance to view the goal of their work on this challenge.

If available class time is an issue, you may want to assign coding the challenge as a homework project and then have the students present their solutions as part of a class discussion.

Applicable CSTA (Computer Science Teachers Association) Standards

2-CS-02	Design projects that combine hardware and software components to collect and exchange data.
2-AP-11	Create clearly named variables that represent different data types and perform operations on their values.
2-AP-12	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

2-AP-13	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
3A-DA-11	Create interactive data visualizations using software tools to help others better understand real-world phenomena.
3A-AP-17	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
3B-CS-02	Illustrate ways computing systems implement logic, input, and output through hardware components.
3B-AP-14	Construct solutions to problems using student-created components, such as procedures, modules and/or objects.