



Lesson

REMEMBERING BRIDGE COLORS

CREATED BY

Richard Born
Associate Professor Emeritus
Northern Illinois University
rborn2@niu.edu

TOPICS

Robotics, Programming

GRADES

7-12

METHOD

OzoBlockly

DURATION

60 minutes

Remembering Bridge Colors with Arrays

By Richard Born

Associate Professor Emeritus

Northern Illinois University

rborn2@niu.edu

Introduction

In this lesson, Bit/Evo will be making multiple crossings on bridges over Ozobot River. The three bridges of interest, shown in Figure 1, are the red, green, and blue bridges. Ozobot wants to test his memory to see if he can remember the color sequence of his multiple random bridge crossings. He will do this *after* he completes all of the crossings, by displaying the sequence of colors in order from the colors saved in his memory.

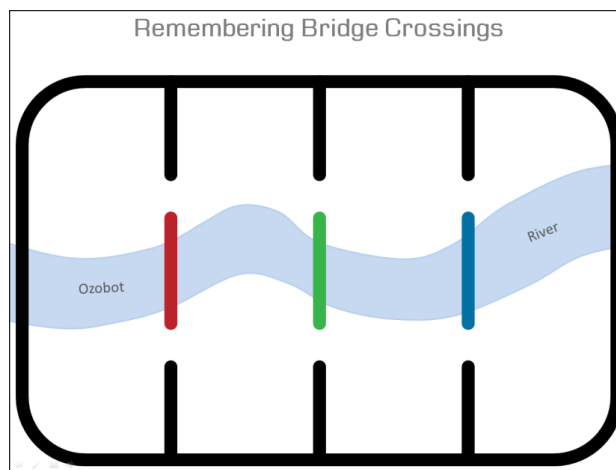


Figure 1

A convenient data structure to accomplish this is the **array**—a group of memory locations with a single name and distinguished from one another by integer indices starting at [0]. The OzoBlockly program consists of a fair number of blocks, so breaking it down into several easy to follow functions will make the logic clearer. As a result, this lesson is good practice for students in modularized program design.

The Video Accompanying this Lesson

The student is now encouraged to view the video, *Remembering Bridge Colors* <https://youtu.be/4iKLP2qcoac>. Doing so will help to clarify precisely what the OzoBlockly program running on OzoBot accomplishes. To keep the video reasonably short, Ozobot has been instructed to cross color bridges a total of only eight times. Ozobot begins traveling with a white LED. At each intersection of the long black border line with a short black line, Ozobot randomly goes straight or turns onto the short black line. After crossing a bridge, Ozobot momentarily displays its top LED to the color of the bridge that it just crossed. After crossing random bridges a total of eight times, Ozobot stops and blinks the top LED in the same sequence of colors in which he encountered the bridges. In the video the sequence is BRRRBGBG. Ozobot then stops and turns off.

Array Declaration and Main Program

Figure 2 shows the OzoBlockly blocks for array declaration and the main program.

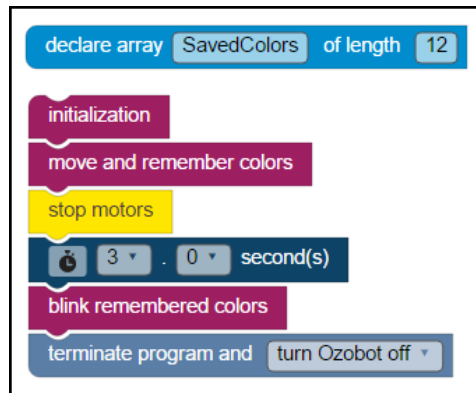


Figure 2

The array *SavedColors* stores the ordered sequence of colors encountered by Ozobot while crossing the bridges over the river. Recalling that the first element of an array has a subscript of [0], then the color of the first bridge crossed is saved in *SavedColors*[0], second bridge crossed in *SavedColors*[1], etc. With the length of the array declared as 12, then the subscripts range from [0] to [11]. So a maximum of 12 bridge crossing colors can be stored in our program. With the maximum allowable length of an array being 127, Ozobot could cross the bridges a total of 127 times and remember all of the colors if we declared the length of the array to be 127.

The main program begins by calling an initialization subroutine, to be discussed in the next section of this document, which does some housekeeping things before Ozobot begins moving. Next, a subroutine is called in which Ozobot begins his journey of visiting the bridges and remembering their colors. Then Ozobots motors are stopped, and after waiting three seconds, he begins to blink the colors of the bridges in the order in which they were encountered.

The “Initialization” Subroutine

Figure 3 shows the OzoBlockly blocks that make up the *initialization* subroutine. This subroutine’s primary purpose is to set the values for several variables in one place in the program, so they can be easily and quickly adjusted later if desired. The author found that the fastest possible speed of 85 mm/s seemed to work quite well for both Bit and Evo. The maximum number of crossings of the river on colored bridges by Ozobot is set to 8. With the array *SavedColors* declared to be of length 12, the largest allowable value for the maximum number of crossings is 12. We need to come up with a convention for representing colors stored in our array. Red will be represented by the integer 1, green by 2, and blue by 3. The variables *red*, *green*, and *blue* are set to these values, so we can reference variables rather than the digits 1, 2, and 3 throughout the program where these digits are needed. The number of crossings is initialized at 0 (and will be increased later by 1 each time Ozobot crosses a colored bridge). Finally, the LED is initially set to white for three seconds. This delay allows the student to start Ozobot while holding it and then setting it down anywhere on a black line before it starts moving.

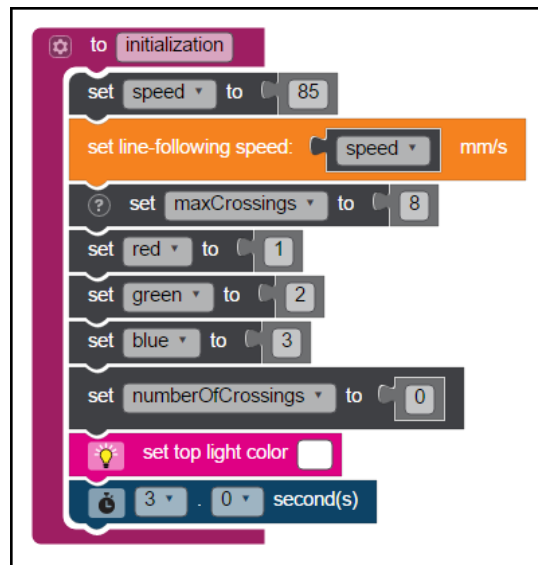


Figure 3

The "Move and Remember Colors" Subroutine

The "move and remember colors" subroutine is the largest subroutine in the program and is shown in Figure 4. At the outermost level of the logic, a *repeat while* loop is executed as long as the number of bridge crossings (*numberOfCrossings*) so far is less than the maximum number of crossings allowed (*maxCrossings*). This loop begins when *numberOfCrossings* is 0 and is incremented by 1 after each crossing. The loop is therefore executed *maxCrossings* times (8 for our program). Within the loop Ozobot is asked to set the top LED color to white and follow the line to the next intersection or line end. (If there is an *intersection*, then Ozobot executes his default of randomly choosing to go straight or turn.) If there is a line end, then he must have just gotten to the end of one of the short black line segments. In this case, he freely moves forward (yellow block) until a (colored) line is found and then follows the line. If the line-end color is red, then another subroutine is called to save the color red and do several other things, to be discussed in the next section of this paper. You will note that the "save color with" subroutine has four input parameters: the color number (1 for red) and the three RGB arguments for displaying red (127, 0, and 0, respectively). Similar actions occur if the line-end color is green or blue.

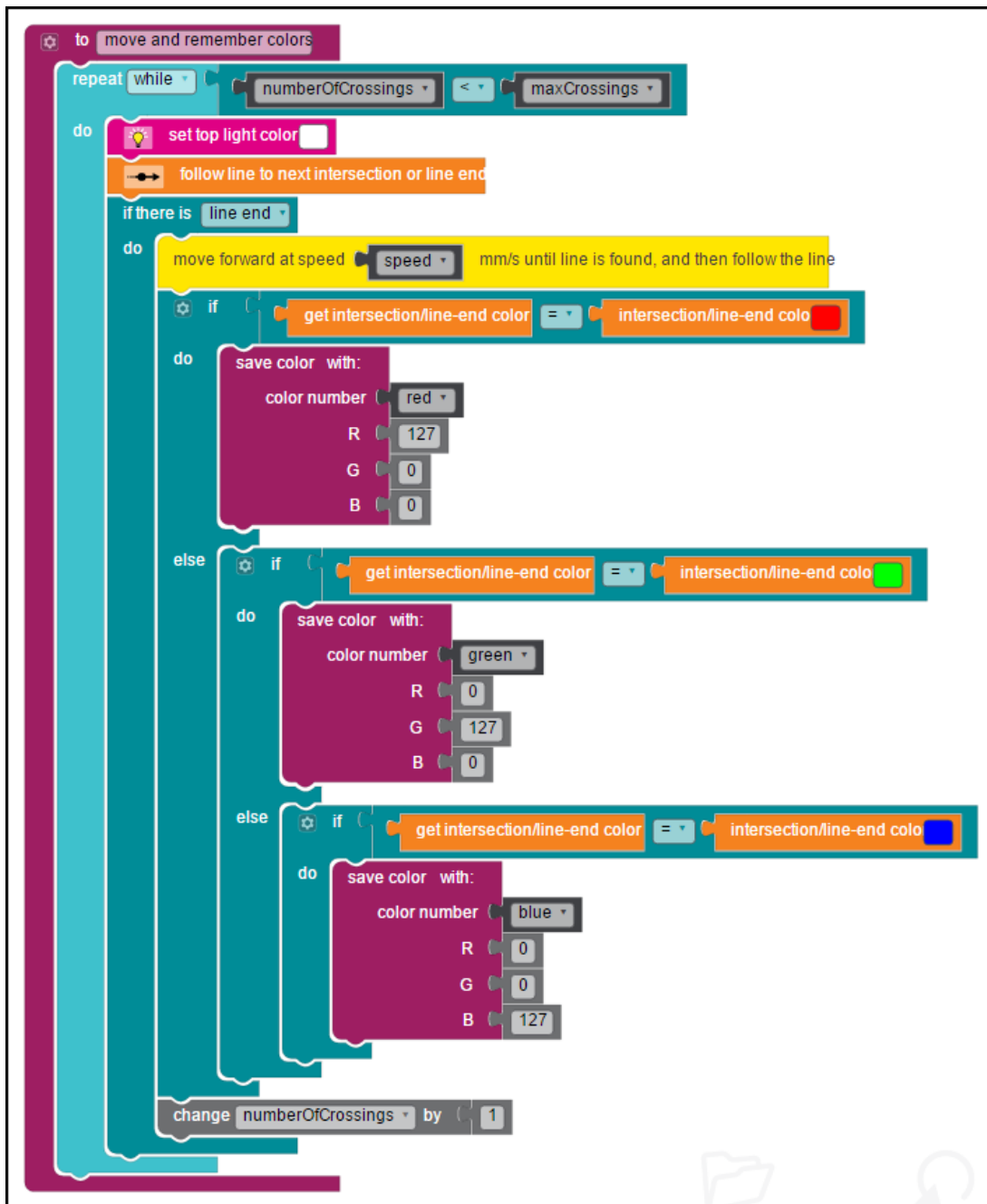


Figure 4

The "Save Color with" Subroutine

Figure 5 shows the blocks in the "save color with" subroutine. The first thing that this subroutine does is save the color number in the element in the *SavedColors* array whose subscript is *[numberOfCrossings]*. The top light color is then set to the color specified by the RGB inputs to the subroutine. Ozobot then freely moves forward until the short black line segment is found. Upon reaching the end of the short black segment on the long black border, the top LED is then set back to white.

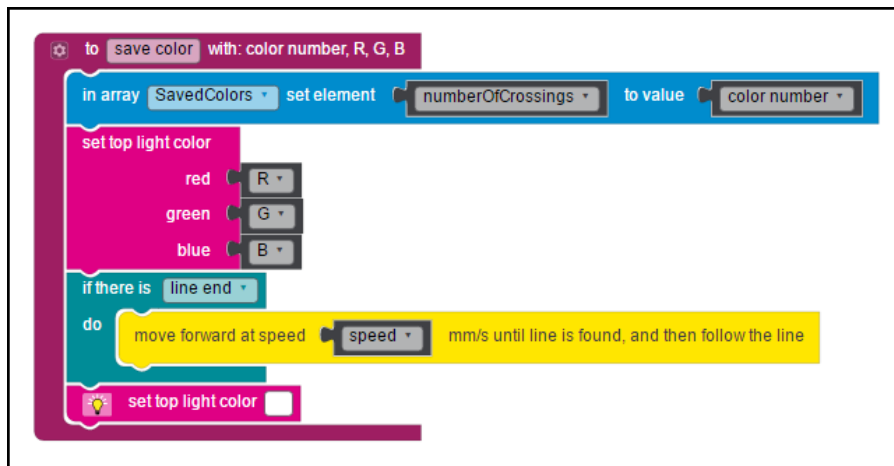


Figure 5

The “Blink Remembered Colors” Subroutine

Figure 6 shows the blocks comprising the “blink remembered colors” subroutine. It begins by turning off the LED for a half-second before blinking the remembered colors. A loop is then executed once for each of the colored bridge crossings. The loop control variable i is also used as the index into the array *SavedColors*. Within the loop, the value of the variable *currentColor* is obtained from the i ’th element of the array. This variable will have the value 1, 2, or 3, representing red, green, and blue, respectively. A pair of nested if blocks then sets the top LED to the color red, green, or blue, as appropriate. The LED stays on for 1 second, and is then off for a half-second.

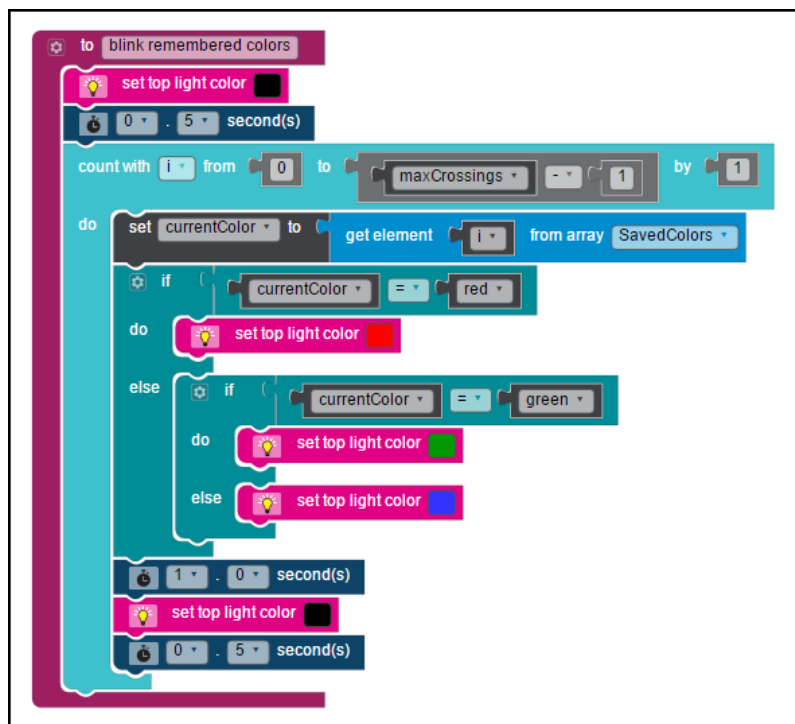


Figure 6

Classroom Activity

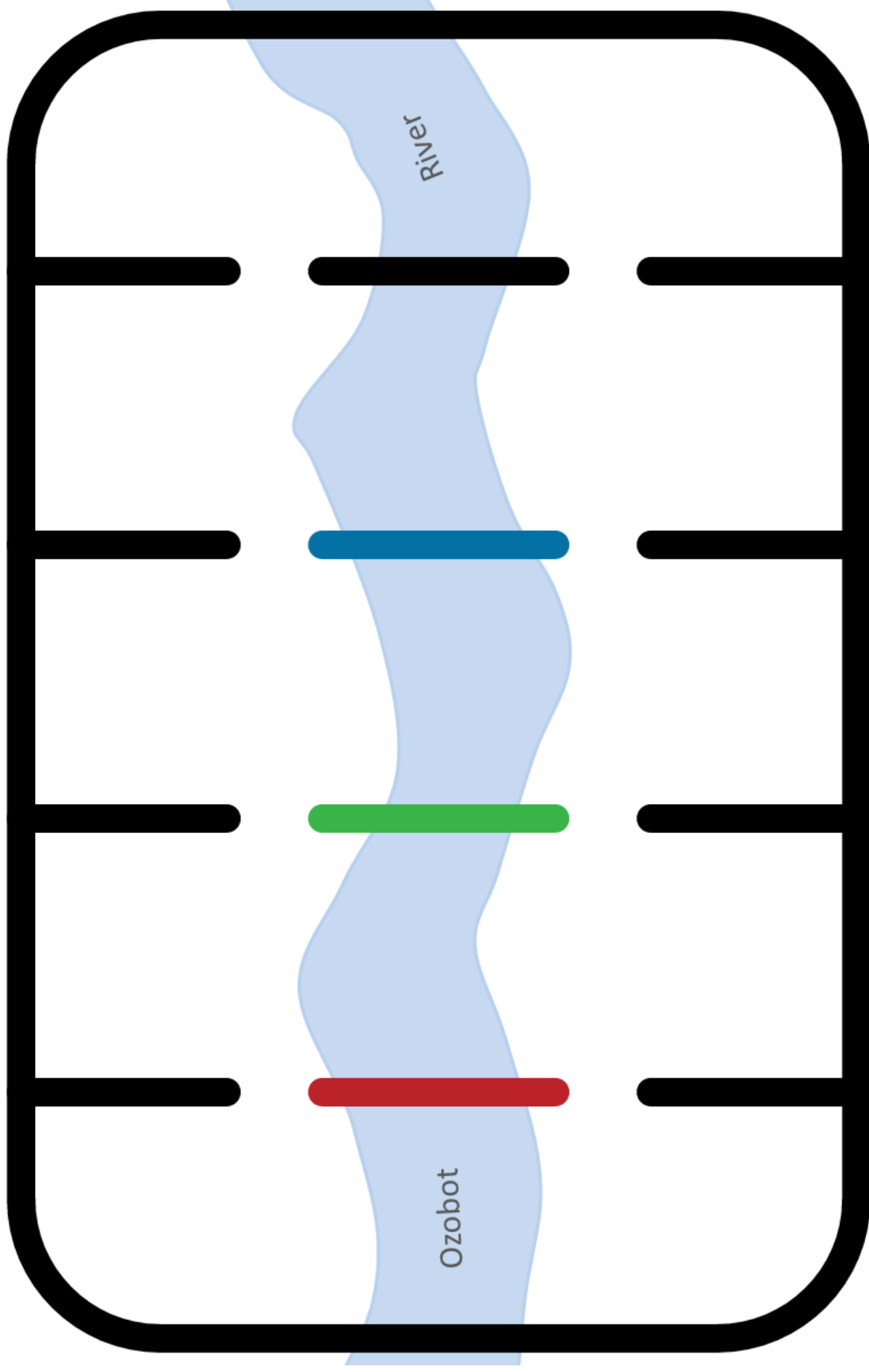
After discussing the OzoBlockly program with the class, provide each student or student group with a copy of the *RememberingBridgeCrossings.ozocode* file and the Ozomap (with the three red, green, and blue bridges) on the final page of this document. Let the students load the program onto their Ozobots and play with it for a few minutes.

Then give them a program maintenance task as follows:

OzoTown has completed the construction of a fourth bridge, but this one is a black bridge. You can see the bridges on the Ozomap entitled “Remembering Bridge Crossings with a New Black Bridge”. Modify the program so that it will also remember crossings of the new black bridge. It should display a yellow LED momentarily after crossing the black bridge. It should blink a yellow LED representing a black bridge when displaying the sequence of bridge crossing colors after all of the crossings are complete. It should perform a combined total of 14 crossings of the four bridges.

The Ozomap with four bridges can be found on page 7 and may be duplicated for student groups.

Remembering Bridge Crossings with a New Black Bridge



Remembering Bridge Crossings

