OzoBlockly Lesson: Evo Stopwatch
A Lesson on Button Press and the Binary Number System

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

Introduction

In this lesson, students turn Evo into a handheld stopwatch capable of timing up to 127 seconds with second precision. In order to accomplish this, students learn the fundamentals of several Mode 5 (Master) level OzoBlockly blocks

The student will work extensively with setting light color for the top light and five front lights on Evo. It is these lights that will display the time that has elapsed since the Evo stopwatch has been started. This time is displayed as a binary number with specific front lights either 1 (on) or 0 (off), depending on the elapsed time. Students therefore gain a detailed understanding of the binary number system, which forms the basis of how digital computers work.

Evo Display of Binary Numbers

Figure 1 shows a close-up of Evo with all five front lights on. Just as the decimal number system place value moves up by a power of 10 for each digit as you move from right to left, the binary number system place values move up by powers of two: $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, and $2^4 = 16$. So the place values for the Evo lights in our program, as you move from right to left, are 1, 2, 4, 8, and 16. Also, just as the *decimal* number system has *ten* symbols for digits (*i.e.*, 0 through 9), the *binary* number system has *two* symbols, *i.e.*, 0 and 1. If we agree that ON means 1 and OFF means 0, then the number shown in Figure 1 is $11111_{TWO} = 1 + 2 + 4 + 8 + 16 = 31_{TEN}$.

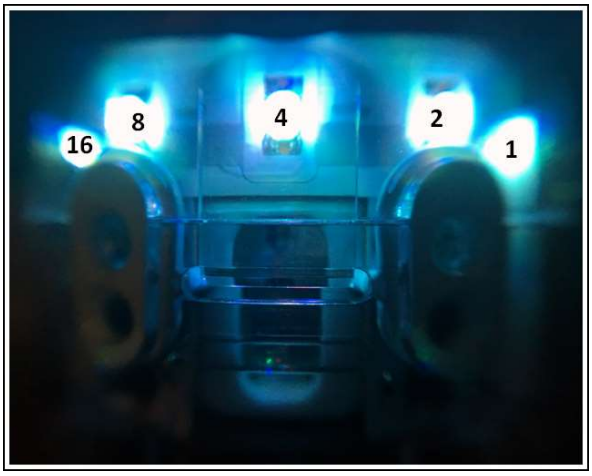

Figure 1

Figure 2 shows the binary representations of decimal numbers from 0 through 31, as they would be expressed by Evo's front lights in our program.

Decimal	***Binary***				
	16	***8***	***4***	***2***	***1***
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1
20	1	0	1	0	0
21	1	0	1	0	1
22	1	0	1	1	0
23	1	0	1	1	1
24	1	1	0	0	0
25	1	1	0	0	1
26	1	1	0	1	0
27	1	1	0	1	1
28	1	1	1	0	0
29	1	1	1	0	1
30	1	1	1	1	0
31	1	1	1	1	1

Figure 2

If you start at the top of a column and move your eye down, you notice some interesting things, *all of which relate to the algorithm we will use to set the colors of Evo's front lights*. For example, if you look at the 1's place column, you notice that 0's and 1's alternate—0 if the decimal number is even and 1 if it is odd. The corresponding Evo front light should be ON if the decimal number is odd. (Alternatively, we could say that the 1's place Evo light should be ON only if the remainder when the decimal number is divided by 2 is 1.)

If you move your eye down the 2's place column, you see *pairs* of 1's followed by *pairs* of 0's. Take a minute or so to convince yourself that when the remainder of the decimal number divided by ***4*** is greater than 2, then the 2's place is 1 and the corresponding Evo light should be on. Otherwise, it should be off.

For the 4's, 8's, and 16's place columns, you would need to look at the remainders when the decimal number is divided by 8, 16, and 32, respectively.

At this point you may be wondering how we can time up to 127 seconds when the Evo front lights only allow timing up to 31. This is where Evo's top light comes into play. We will start with the top light red as Evo counts from 0 to 31. When Evo gets to 32 and the front lights are all off, we will change the color of the top light to green. Therefore, whenever the top light is green, we need to add 32 to the value represented by the front lights. When Evo gets to 2*32 = 64, we will change the color of the top light to blue. In this case we need to add 64 to the representation of the front lights. When Evo get to 3*32 = 96, we will change the top light to yellow. In this case, we need to add 96 to the representation of the front lights. *We might call this our **RGBY rule**—**R**ed add 0, **G**reen add 32, **B**lue add 64, and **Y**ellow add 96.* Four examples for studying are shown in Figure 3.

Top Light	***Front Lights***					***Timer in Seconds***
	16	8	4	2	1	
Yellow	ON	OFF	ON	OFF	OFF	96 + 20 = 116
Blue	OFF	ON	OFF	ON	OFF	64 + 10 = 74
Red	ON	OFF	OFF	OFF	ON	0 + 17 = 17
Green	ON	ON	OFF	ON	ON	32 + 27 = 59

Figure 3

Instructions for Running the OzoBlockly Evo Timer

1. Load the program *EvoStopwatch.ozocode*, provided in the accompanying files, onto Evo. You can find details for loading Evo at the ozoblockly.com Web site.
2. With Evo turned off, press the start button once to turn Evo on. The front lights will flash light blue for a second and Evo will make a sound.
3. Then ***quickly*** after step 2, press the start button twice to start the stopwatch program. The top light will turn white. It will remain white until you perform step 4.
4. To start the stop watch, press Evo's button once. It will begin counting seconds in binary as discussed in the previous section of this document.
5. To stop the stop watch, press Evo's button once. It will continue to display the elapsed time until you perform step 6. The discussion in the previous section explains how to read the elapsed time.
6. Press the button once to terminate the program and turn Evo off.

Classroom Activity

After discussing the OzoBlockly program with the class, provide each student or student group with a copy of the *EvoStopwatch.ozocode* file and let them load the program onto their Evos and play with the program for a while. Then have them perform the following activities:

1. After the students are confident in working with the program, ask them to use their Evo stopwatch to time 25 seconds on their wrist watches or the classroom wall clock. Does the Evo stopwatch agree well with their wrist watch or classroom wall clock?

2. The teacher could, for example, ask students to measure the time from when she flips on a classroom light until she turns it off again. The students would then need to convert the binary time into decimal

seconds. Are the results reasonably consistent from one student group to another? If a group is way off from the values of most other groups, why are they so far off?

The OzoBlockly Program

The Evo stopwatch OzoBlockly program consists of a main program and one user-defined function. The blocks making up the main program can be seen in Figure 4. The first block in the program allows the program to capture button press events, disabling the automatic power-off feature when a button is pressed. The top light is set to the color white, and then we have a block that waits for a button press event from the user to start the stopwatch. There is then a *count with* loop for which the control variable *i* varies from 0 to 31, by steps of one. Within this loop, a user-defined subroutine displays the current count *i* on the front lights. The button press count is set to zero, and there is a 1-second delay. A button press, recognized by the button count being at least 1, means that the user wants to stop the stopwatch—accomplished by breaking out of the loop and executing the block immediately after the loop. After breaking out of the loop, one additional user button press turns off the capture of button presses, and then turns Evo off.

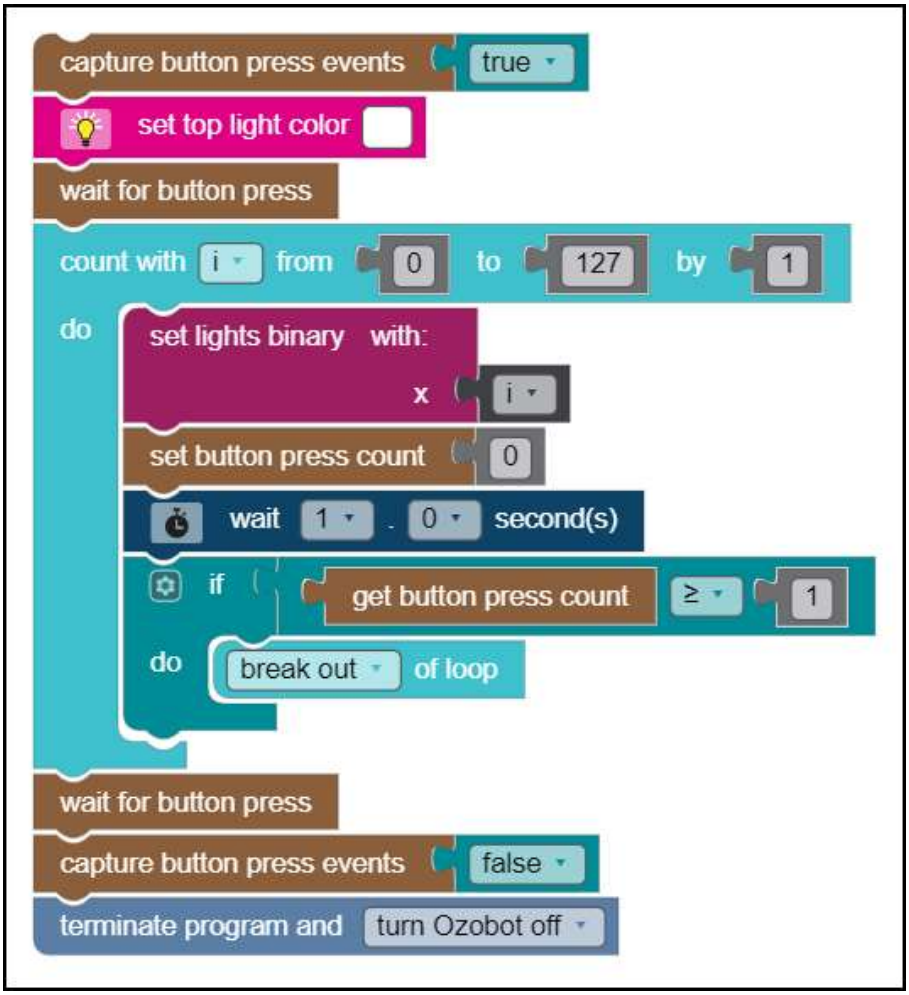

Figure 4

The "set lights binary" subroutine appears in Figures 5(a) and 5(b). Figure 5(a) contains the first part of the subroutine and Figure 5(b) contains the second part. The *if…else if* block in Figure 5(a) deals with the RGBY

rule discussed on page 3 of this document. It is critical to note that the order of the three conditions that are tested is important, with the *most-restrictive* ">" condition checked first. The *set light color* block is used to set all of the front lights OFF, i.e., with the red, green, and blue components all zero. In other words, it resets all of the front lights so they do not show ON for any lights that were on in the previous count displayed. This is necessary as this block does not change the status of any lights that are left unchecked.

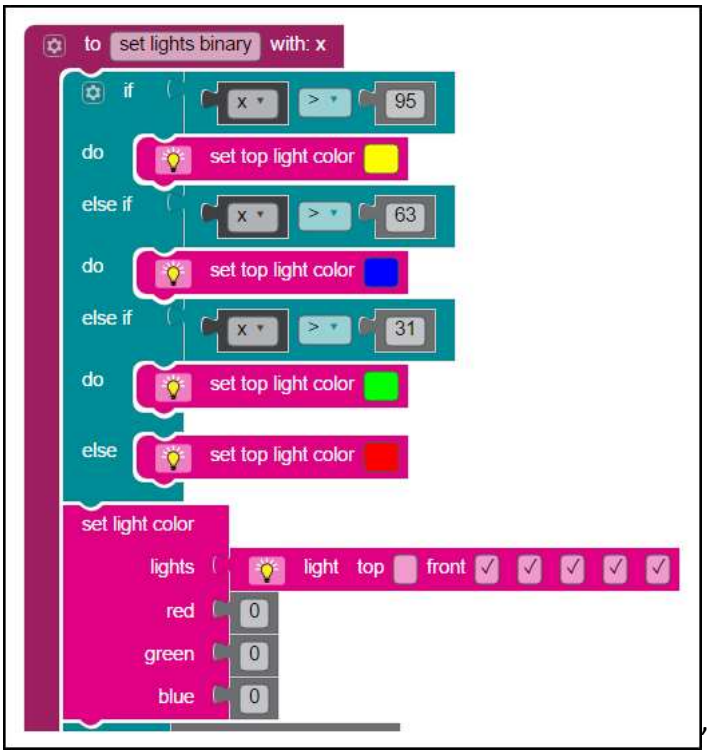

Figure 5(a)

Figure 5(b) contains the blocks that adjust the front lights for each new stop watch second count. Although the figure is quite large, the five *if* blocks are all quite similar in what they accomplish. They deal with the discussion on page 2 on setting the status of each of the front lights, based upon the decimal value of the current second count of the loop variable *i*. In all of these *set light color* blocks, you will note that the red, green, and blue components of the color are all small at the value 8. By keeping the values the same for all three components, we are setting the color to a somewhat dimmed gray. It was the opinion of the author that pure white (127, 127, 127) is too bright for the viewer's eyes.

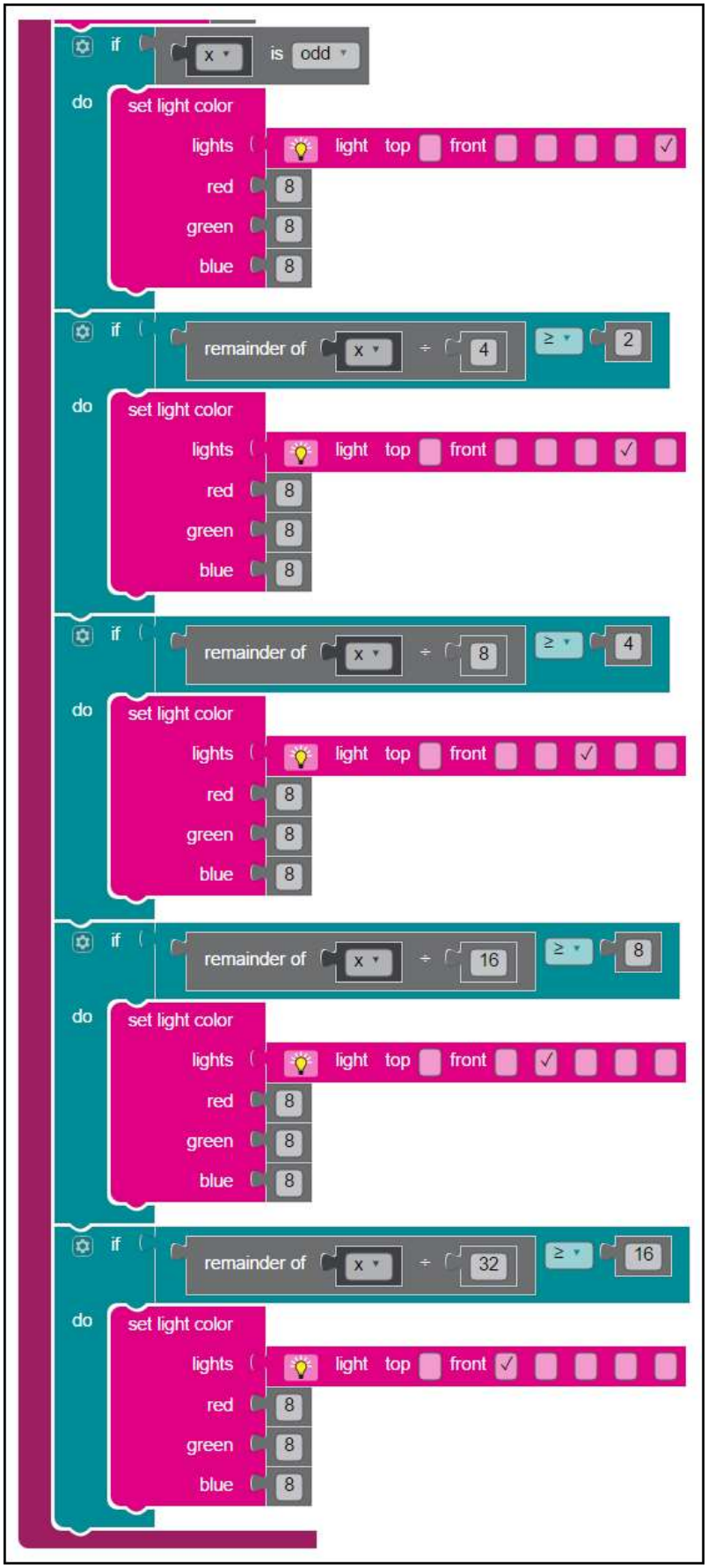
if x is odd
do set light color lights light top front ✓ red 8 green 8 blue 8
if remainder of x ÷ 4 ≥ 2
do set light color lights light top front ✓ red 8 green 8 blue 8
if remainder of x ÷ 8 ≥ 4
do set light color lights light top front ✓ red 8 green 8 blue 8
if remainder of x ÷ 16 ≥ 8
do set light color lights light top front ✓ red 8 green 8 blue 8
if remainder of x ÷ 32 ≥ 16
do set light color lights light top front ✓ red 8 green 8 blue 8

Figure 5(b)