Production Line Simulation and Robotic Preventive Maintenance

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

Introduction

There are dozens, if not hundreds, of robotic toys in the marketplace today. But none of them has the advantage of small size that Ozobot can claim. This small size and its programmability make Ozobot perfect for simulating countless real-world things. In this lesson, six Ozobots (Bit or Evo) are programmed to simulate factory production lines similar to the robotic production lines in so many of today's modern factories. The use of robots has allowed manufacturers to see many industry benefits, but these benefits may be short-lived if they are not accompanied by a robot maintenance program. The predominant type of robot maintenance is *preventative maintenance*, including carefully planned testing, inspections, and activities such as cleaning sensors and optics, lubrication, and part replacement. The primary purpose for all of this is to keep the probability of deterioration, breakdowns, and failure to a minimum.

Figure 1 shows a snapshot of six Ozobots on the provided Ozomap, all of them running the same OzoBlockly program, to be discussed in this document. Each Ozobot is simulating a robot on an automobile production line in a factory. When the top LED is green, the robot is busy working on a car. When the top LED is blue, it is retrieving a part for the next car, and when the top LED is red, Ozobot is in preventive maintenance. A separate accompanying document contains a full size Ozomap suitable for copying for students to use while running the OzoBlockly program.

Figure 1

The Production Line Video

At this time you are encouraged to view the accompanying video, entitled *ProductionLine.m4v*. Doing so will help you to visualize what the OzoBlockly program for this lesson does. The program is loaded onto all six Ozobots, with each on a short horizontal black line facing the conveyor belt that runs vertically up the center of the Ozomap. Each of our robots stops for a short time and shows a green LED while working on the car on the conveyor belt. Ozobot then turns around and displays a blue LED while spending a brief time fetching a part for the next car. Periodically, Ozobot stops for several seconds for preventative maintenance. During this time period, Ozobot shows a red LED. This continues until the Ozobots are turned off or run out of battery charge.

The OzoBlockly Program

This lesson will, among other things, explain how to use the ***timer*** provided in Master Mode (5) of OzoBlockly. The program consists of a main program and several subroutines that break the code down to smaller, more manageable functional pieces. The main program consists of calls to two subroutines, as shown in Figure 2.

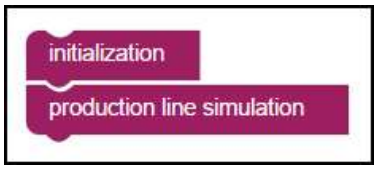

Figure 2

The "Initialization" Subroutine

The initialization subroutine is shown in Figure 3. The first thing we do is initialize parameters for the simulation's random variables. Doing this all in one place and using these variables throughout the program makes it easy to adjust and experiment with the parameter values if desired. Variable names are selected that are as self-documentary as possible. All values are times in seconds. For example, the time required to perform maintenance lies between 7 and 12 seconds on Ozobot's clock. Most simulations compress time, and this is no exception for Ozobot. We assume that real factory wall-clock times are 60 times greater than the times supplied for Ozobot. Therefore, 1 second of the Ozobot clock corresponds to 1 minute of a real wall-clock in the factory. The time to perform maintenance is between 7 and 12 minutes on a real clock in our simulated factory. As seen in Figure 3, the time interval between regular robot maintenances is 60 minutes, or 1 hour. The time that the robot works on a car is between 1 and 3 minutes. Similarly, the time required for the robot to fetch a part is also between 1 and 3 minutes.

The last part of the initialization subroutine sets Ozobot's top light to the color white and then waits 3 seconds, giving time to place Ozobot on the Ozomap before Ozobot starts to move. Finally, Ozobot's timer is set to zero. This timer is used to enforce regularly scheduled maintenance intervals of 1 hour.

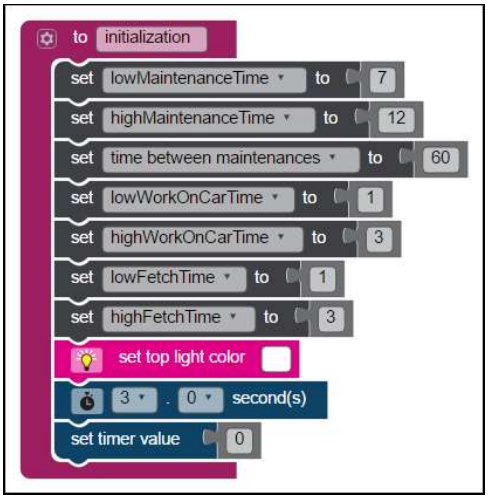

Figure 3

The "Production Line Simulation" Subroutine

Figure 4 shows the blocks comprising the "production line simulation" subroutine. At the outermost level, it consists of a *loop forever* loop for which no *break out of loop* is provided. As mentioned earlier, the simulation will continue until Ozobot is turned off or runs out of energy from the battery.

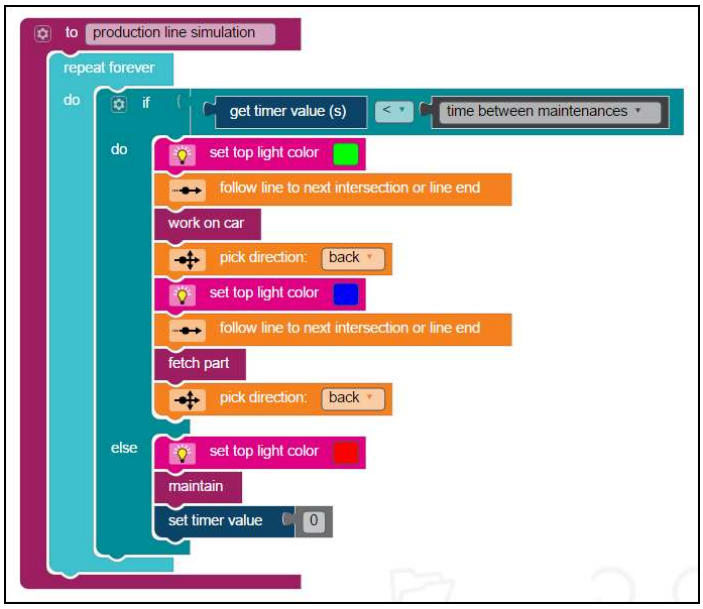

Figure 4

Inside the loop is an *if…do…else* block. This block checks to see if the timer is less than the time between maintenances. If so, it sets the LED green, follows the line to the next intersection, and then works on the car. Then it turns around (direction back), sets the LED blue, follows the line to next intersection, and fetches a part for the next car. After doing this, Ozobot, now at the end of a line, turns around again. This process is repeated until the condition in the *if…do…else* block is false. In this case the blocks in the *else* clause are executed. The LED is set to red, and the robot goes into preventative maintenance mode. After maintenance is complete, the value of the timer is reset to zero to start another maintenance interval.

The "Work on Car", "Fetch", and "Maintain" Subroutines

These three subroutines are all similar in structure. Differences are simply in the random numbers that they produce to represent the wait times for these three robot activities. Figure 5 shows the blocks for these three subroutines. Each of the three subroutines begins by computing a random integer between the low and high values. This random integer is stored in the user-defined variable *rand*. Then a loop is repeated *rand* times, each time with a wait time of 1 second. The result is that Ozobot is at rest for the desired process (either, work on car, fetch part, or maintain) time.

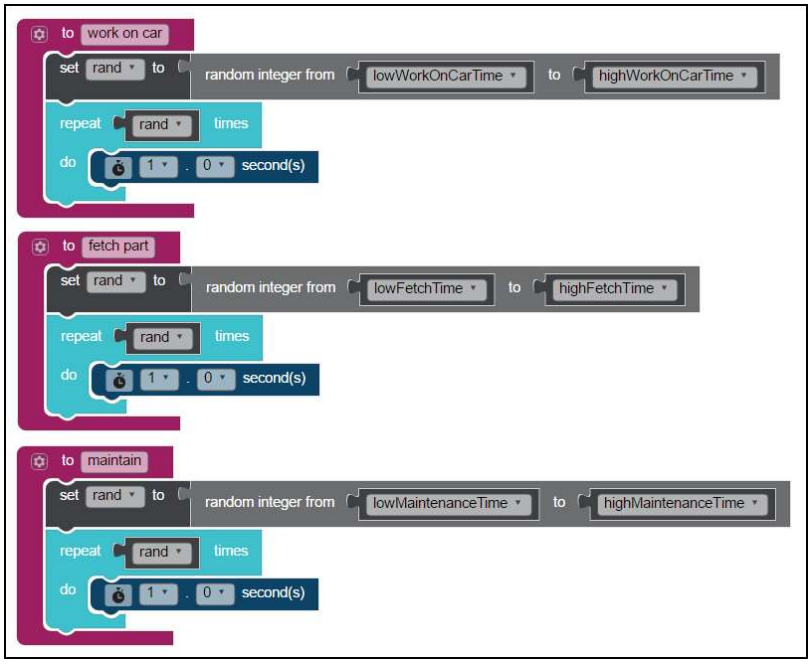

Figure 5

Classroom Activity

You have been given access to a copy of the *ProductionLineSimulation.ozocode* file. Load the program into Ozobot and play with the program for a while. Then perform the following program maintenance task:

> *Immediately before each hourly maintenance operation, the robot is to report the number of cars that it worked on during that hour. This is to be accomplished by having Ozobot blink its top LED white the number of times that it worked on a car. Make use of a user-defined subroutine to accomplish this.* ***Students with Evos should have Evo speak the number of times that it worked on a car instead of blinking its top light****.*