Ozobot Lesson: State-Machines Design and Implementation
Created by Andrei I. Nasushnyi

Ages
Grades 7-12

Duration
90 minutes or more

Method
OzoBlockly

OzoBlockly programming topics
Variables, Movement, Logic

Teacher Materials & Digital Resources

https://drive.google.com/file/d/1FmJUAA6_aeP6jikh9EGqHiweySYLhgpo/view?usp=sharing

https://ozoblockly.com/editor?lang=en&robot=bit&mode=4#32yrnp

https://ozoblockly.com/editor?lang=en&robot=bit&mode=4#3ujngg

Brief Summary

The lesson is to introduce a general review of a state-machine approach and to program the lines following behavior based on the basics of the approach.

Students will be able:

Program and implement line following behavior based on the state-machine approach in OzoBlockly demonstrating the correct work of such solutions;

Analyze the general principle of the state-machines implementation;

Test and troubleshoot written code for the optimum work of the solutions;

Apply knowledge of logic and reinforce the usage of variables for programming of the line following behavior.

Procedure

Explain to students that they will use a state-machine design pattern to program the lines following behavior. Demonstrate the work of the finished program so that students can familiarize themselves with the lesson goal. Solutions are included in this lesson.

To fulfill the purposes of the lesson pupils need to be introduced to the basic theory of a finite-state machine.

Basically, a state-machine can be described as a simplified model of a certain system, which has a finite set of states and conditions of transitions leading one state to another.

To illustrate the explanations, ask pupils to draw a line following track consisting of two colors. Explain that the solution of an autonomous behavior will be based on the use of the on-off algorithm. There are two possible reactions to the colors we can have, namely: turn left or right.

Figure 1. An example of a line following track.

Ask pupils how many states they think their robots can have to follow the path.

We can identify two different states in the robot's behavior: to move right or left.

The teacher can help by compiling the state-transition table which describes the introduced state machine.

State-Transition Table		
State	Input	Output
State 1(Turn Right)	Red	Right motor = 50 Left motor = 15
State 2 (Turn Left)	Green	Right motor = 15 Left motor = 50

Table 1. The State-Transition Table filled out for the figure 1 track.

All possible states and inputs are enumerated across the rows of the table. This table shows what state our state machine will move to, based on the current input from the sensors. When our robot receives a red sensor value the state machine will transition or stay in state 1 and vice versa.

To elaborate, you can also draw or show the state diagram.

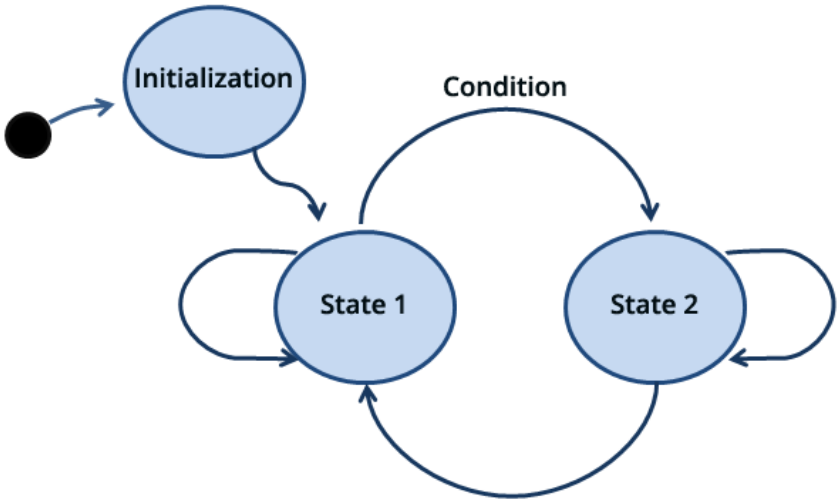

Diagram 1. The State Diagram corresponding to the discussed state machine.

Here two nodes or simply circles represent each state given. Arrows show the transitions between states. There are two possible inputs that can trigger that transition condition: detected red or green color. When the input does not cause a change of state it is denoted by the arrow looping the original state. The independent circles are designated to reproduce the formal structure of a finite-state machine. The black node indicates the start of the program. The following circle is an initialization state designated for the initial setting of the program. The state machine described above is deterministic because the system transitions from one state to another.

With that abstract description of the system behavior in mind, we can finally start building the first program.

Guide your students through the programming process.

Create and name three basic structural functions, they are “Initialization”, “Action Update”, and “State Update” functions.

The “Initialization” function is a place where we put in all needed variables and make a current state sensor reading. There should be a variable to store the current state of our system and variables for color inputs.

The “State Update” function defines the transitions in the program, this part of the program answers for the logic of state change. Here is where sensors’ reading happens to provide a series of transitions.

The “Action Update” function is the functionality code; it allows our robot to move accordingly to its sensors’ reading. Here we could create two more functions for two of our states.

To support pupils display the finished solution.

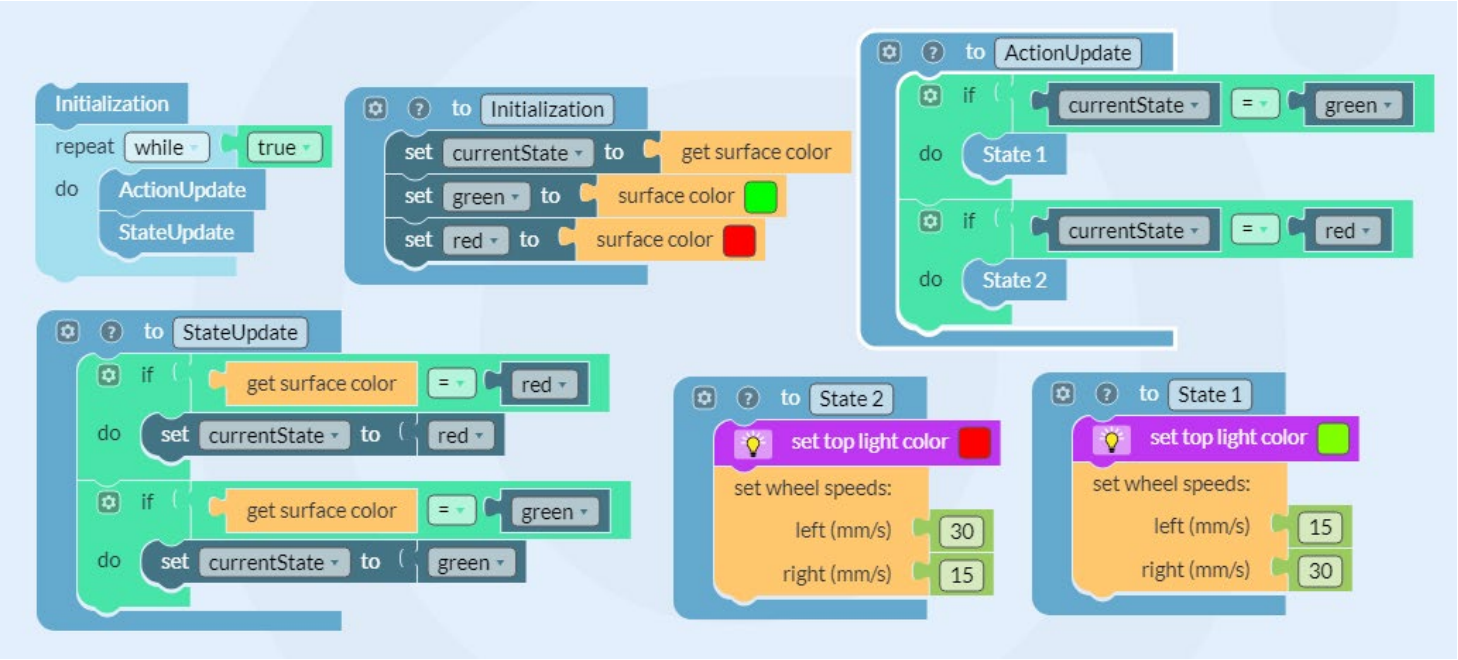

Figure 2. Sample code

Explain to the students that even though the state machine programming style could seem a complicated way to program such a simple controller, it is easy to scale and make more complex behavior.

Have your students make the program to line follow even more colors and demonstrate more complex autonomous behavior. Demonstrate the map they could copy to draw their maps.

Figure 3. An example of a line following track.

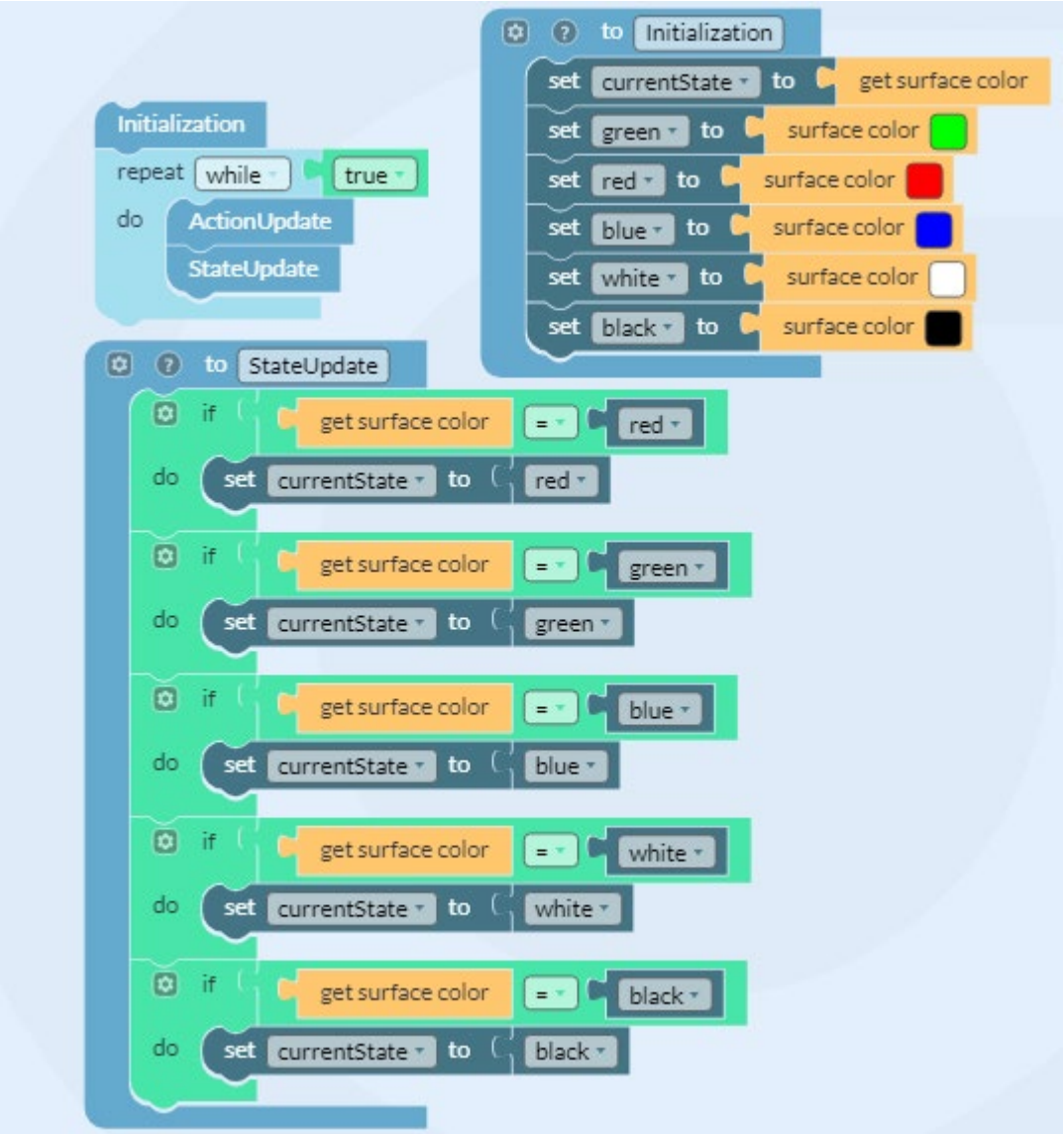

Figure 4. Solution 2 part 1

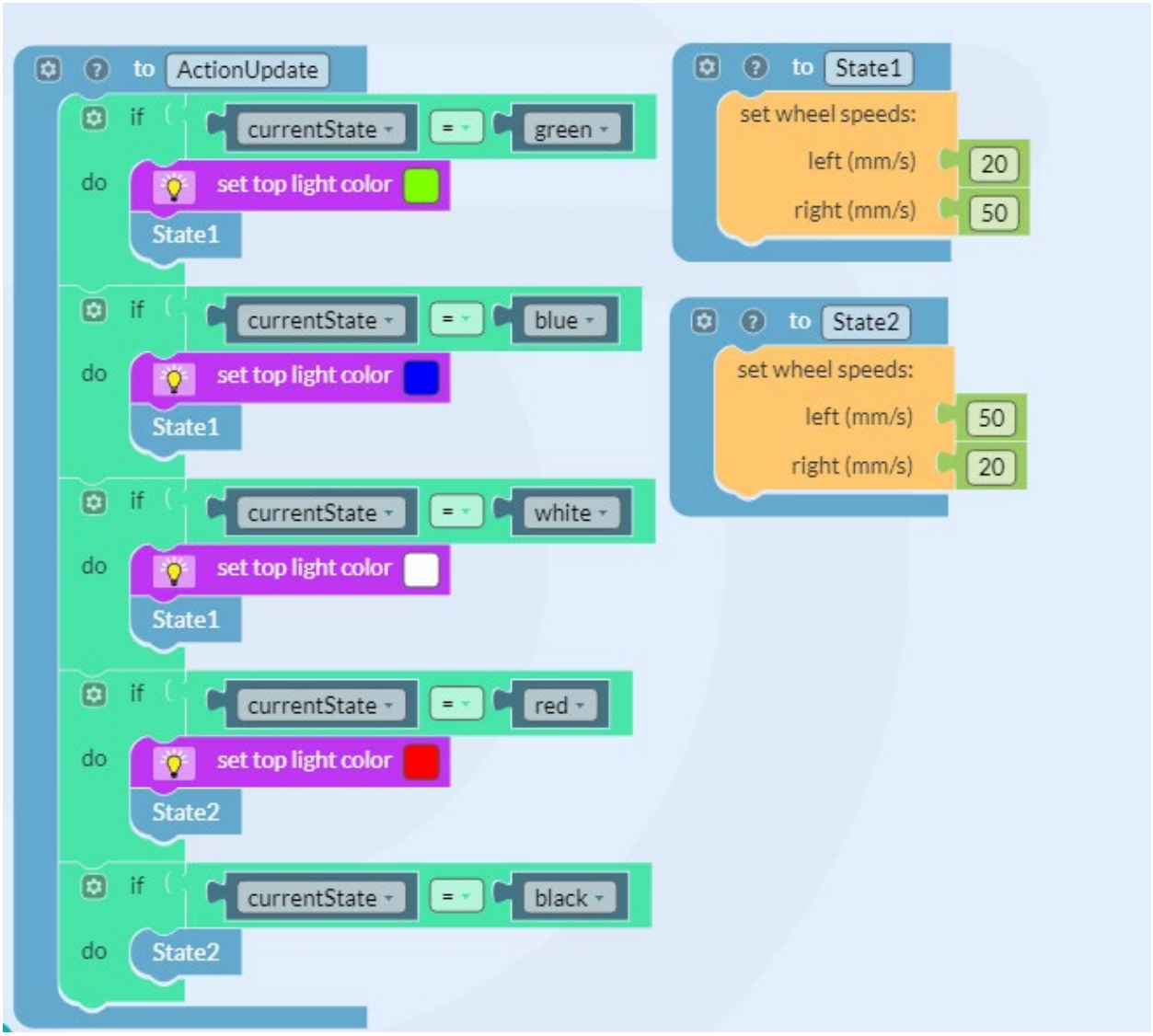

Figure 5. Solution 2 part 2

Confer with students. Review the lesson material.

Note for students

Your teacher will provide all needed guidance and instructions to successfully fulfill the purposes of the lesson.