ADVANCED EV3 PROGRAMMING LESSON

Proportional Control

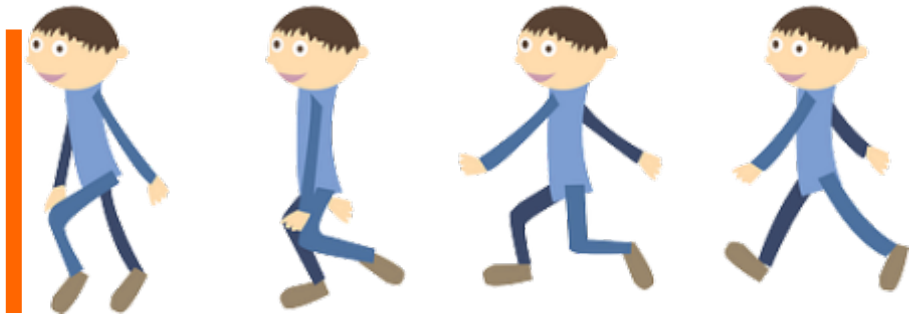
By Sanjay and Arvind Seshan

Lesson Objectives

- Learn what proportional control means and why to use it
- Learn to apply proportional control to different sensors
- Prerequisites: Math Blocks, Color Sensor Calibration, Data Wires

Learn and Discuss Proportional Control

- On our team, we discuss “proportional” as a game.
- Blindfold one teammate. He or She has to get across the room as quickly as they can and stop exactly on a line drawn on the ground (use masking tape to draw a line on the floor).
- The rest of the team has to give the commands.
- When your teammate is far away, the blindfolded person must move fast and take big steps. But as he gets closer to the line, if he keeps running, he will overshoot. So, you have to tell the blindfolded teammate to go slower and take smaller steps.
- You have to program the robot in the same way!

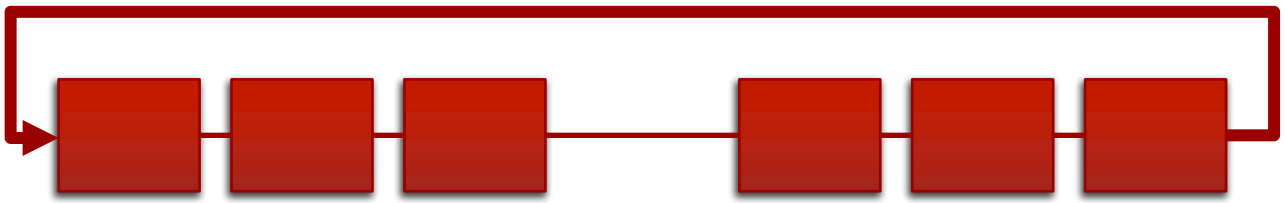


Why Proportional Control?

- What does proportional mean?
  - The robot moves proportionally – moving more or less based on how far the robot is from the target distance
  - For a line follower, the robot may make a sharper turn if it is further away from the line
- Proportional Control can be more accurate and faster
- The Pseudocode for every proportional control program consists of two stages:
  - Computing an error → how far is the robot from a target
  - Making a correction → make the robot take an action that is proportional to the error (this is why it is called proportional control). You must multiply the error by a scaling factor to determine the correction.

What Proportional Control Looks Like

- The Pseudocode for every proportional control program consists of two stages:
  - Computing an error → how far is the robot from a target
  - Making a correction → make the robot take an action that is proportional to the error (this is why it is called proportional control). You must multiply the error by a scaling factor to determine the correction.

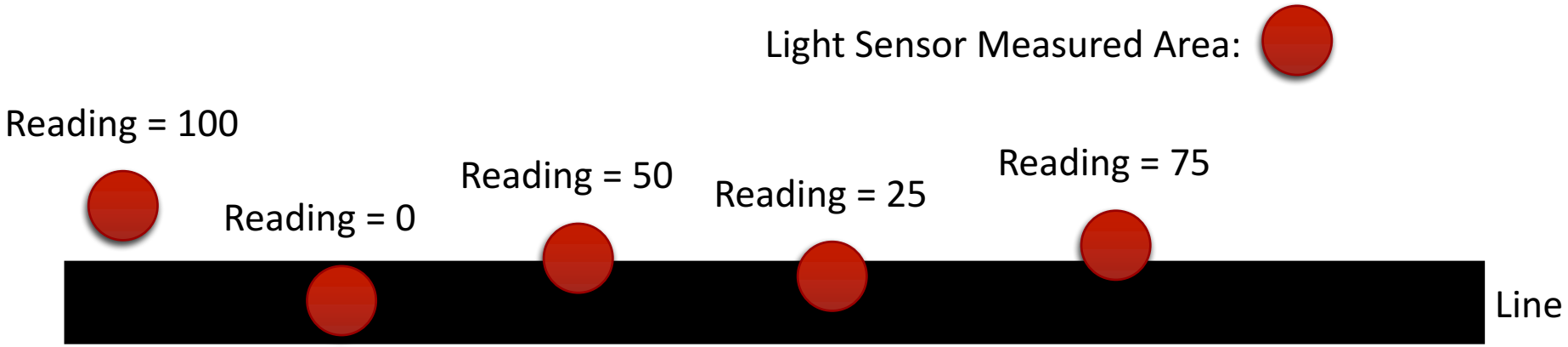


Compute Error

Make Correction

How Far Is the Robot From The Line?

- Reflected light sensor readings show how “dark” the measured area is on average
- Calibrated readings should range from 100 (on just white) to 0 (on just black)



Line Following

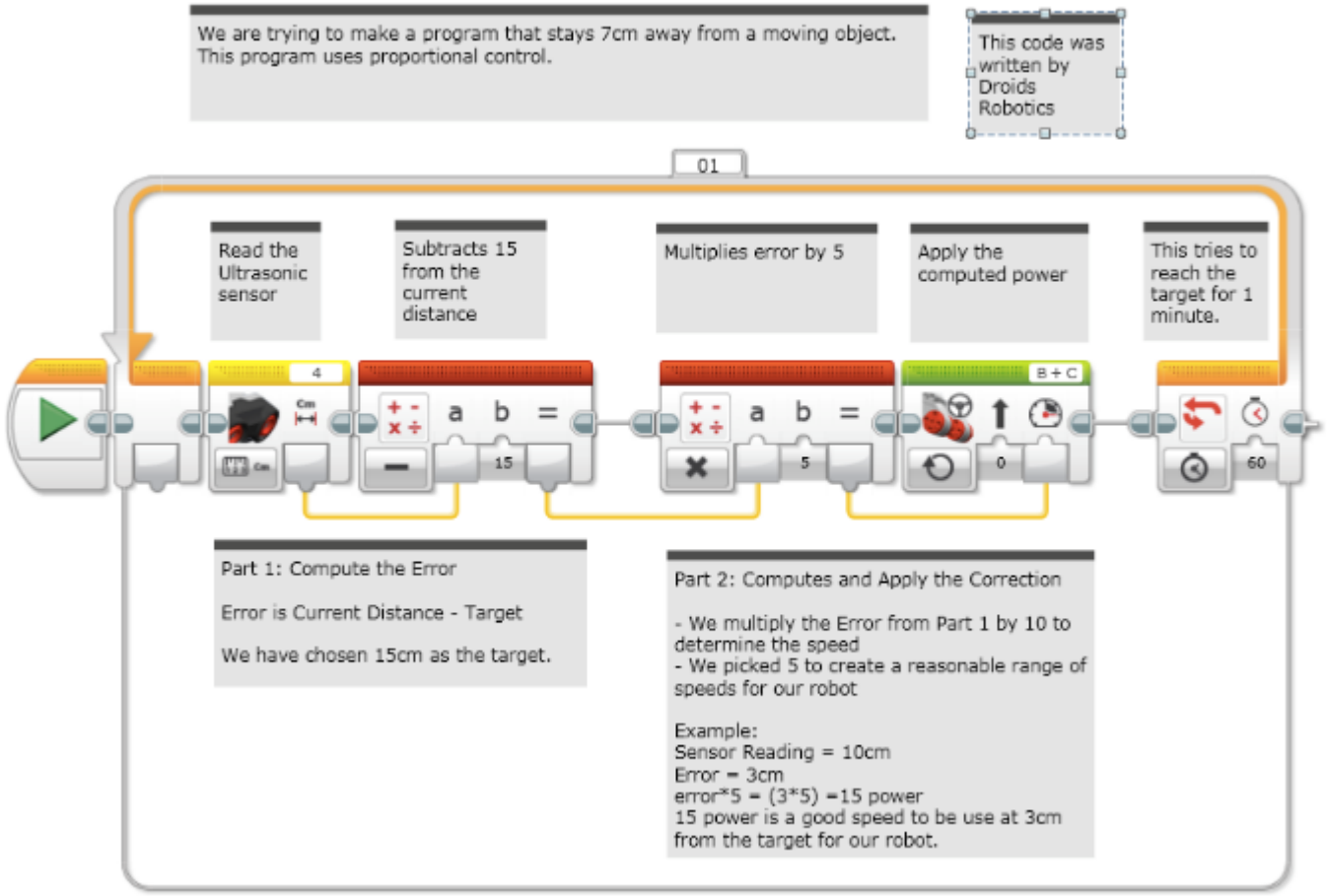
- **Computing an error** → how far is the robot from a target
  - Robots follow the edge of line → target should be a sensor reading of 50
  - Error should indicate how far the sensor's value is from a reading of 50
- **Making a correction** → make the robot take an action that is proportional to the error. You must multiply the error by a scaling factor to determine the correction.
  - To follow a line a robot must turn towards the edge of the line
  - The robot must turn more sharply if it is far from a line
  - How do you do this: You must adjust steering input on move block

Challenge

- To learn how to use proportional control, create a Dog Follower program
  - Use proportional control with the ultrasonic sensor to get the robot to stay 15cm away from the human at all times (even when the human moves)

Application	Objective	Error	Correction
Dog Follower	Get to a target distance from wall	How many inches from target location (current_distance – target_distance)	Move faster based on distance

Solution: Ultrasonic Dog Follower



Discussion Guide

1. What does proportional control mean?
   Ans. Moving more or less based on how far the robot is from the target distance

2. What do all proportional control code have in common?
   Ans. Computing an error and making a correction

Credits

- This tutorial was created by Sanjay Seshan and Arvind Seshan
- More lessons at www.ev3lessons.com