ADVANCED EV3 PROGRAMMING LESSON

Proportional Control

By Droids Robotics with code from the Construction Mavericks

Lesson Objectives

- Learn what proportional control means and why to use it
- Learn to apply proportional control to the Gyro, Color, and Ultrasonic Sensors

- Prerequisites: Math Blocks, Color Sensor Calibration, Data Wires

Learn and Discuss Proportional Control

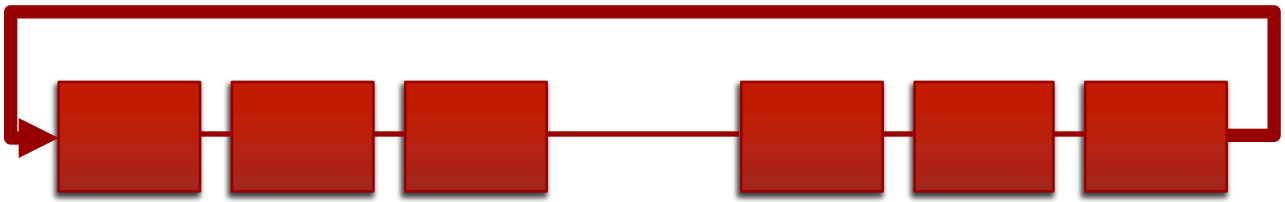
- On our team, we discuss “proportional” as a game.
- Blindfold one teammate. He or She has to get across the room as quickly as they can and stop exactly on a line drawn on the ground (use masking tape to draw a line on the floor).
- The rest of the team has to give the commands.
- When your teammate 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.

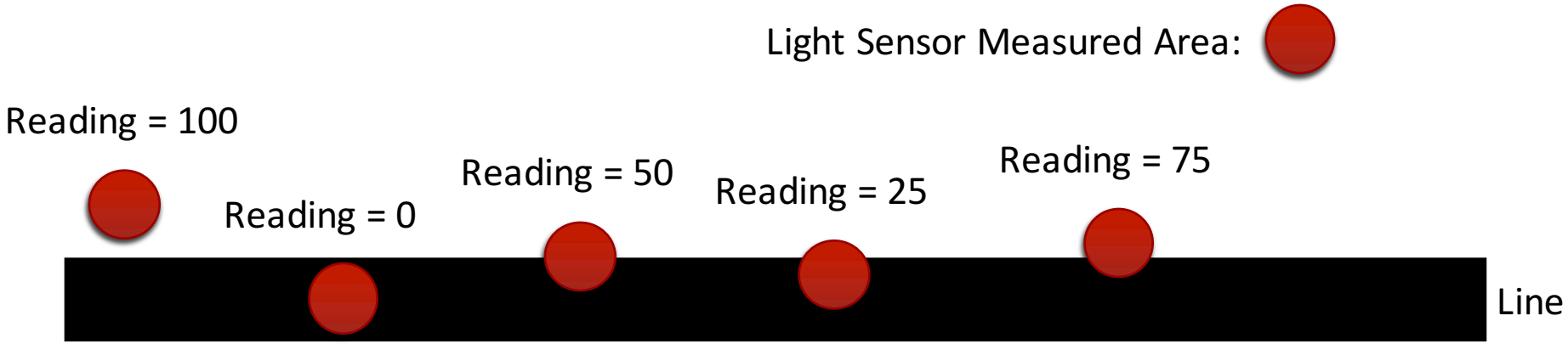


Compute Error

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

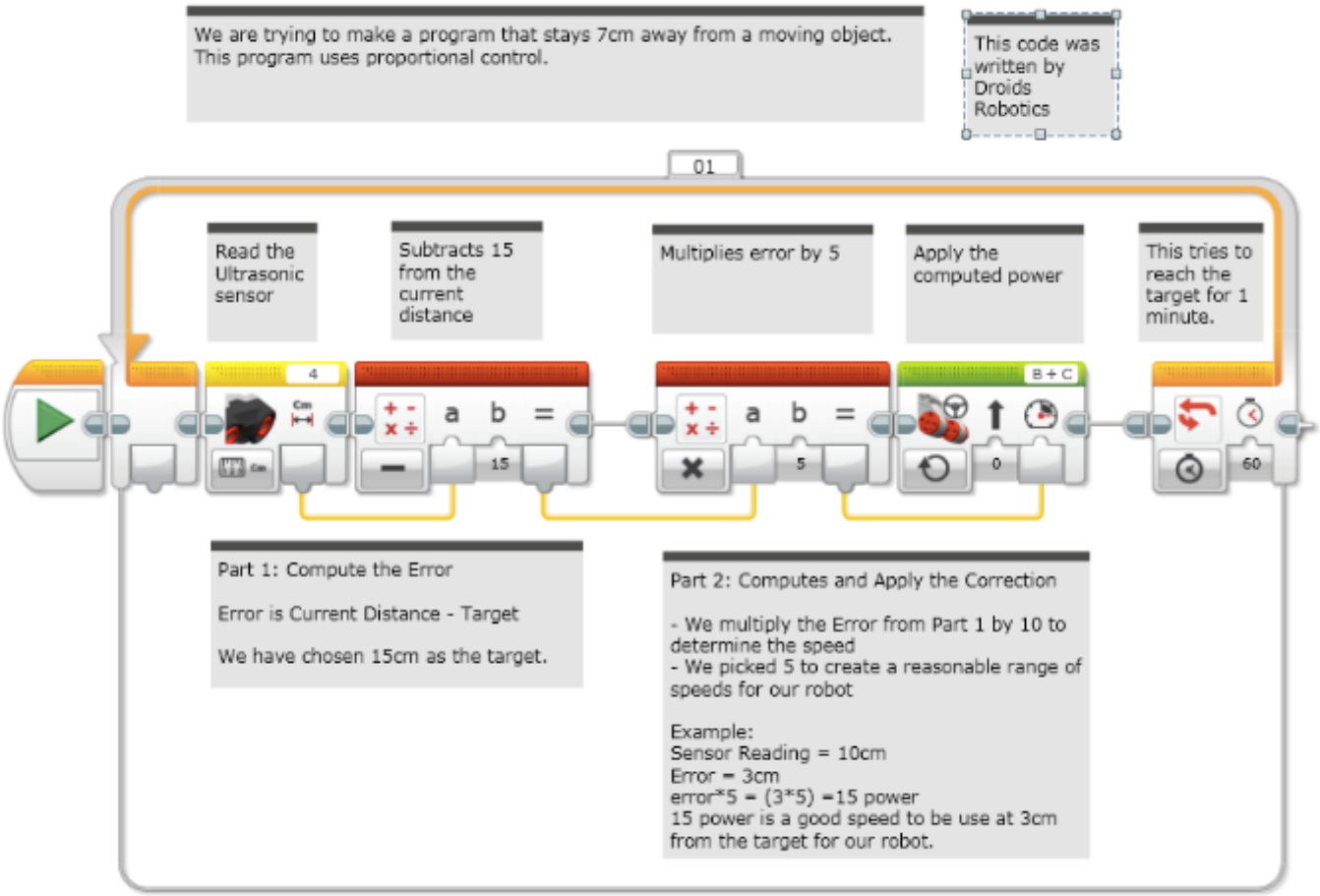
Challenges

- To learn how to use proportional control, we give you three different Challenges:
  - Dog Follower: 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)
  - Line Follower: Use proportional control with the light sensor to get the robot to follow a line smoothly. (Greater detail is in the Proportional Line Follower lesson)
  - Gyro Turn: Use proportional control and the gyro sensor to get the robot to accurately turn to a target angle

Pseudocode/Hints

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
Line Follower	Stay on the edge of the line	How far are our light readings from those at line edge (current_light – target_light)	Turn sharper based on distance from line
Gyro Turn	Turn to a target angle	How many degrees are we from target turn	Turn faster based on degrees remaining

Solution: Ultrasonic Dog Follower

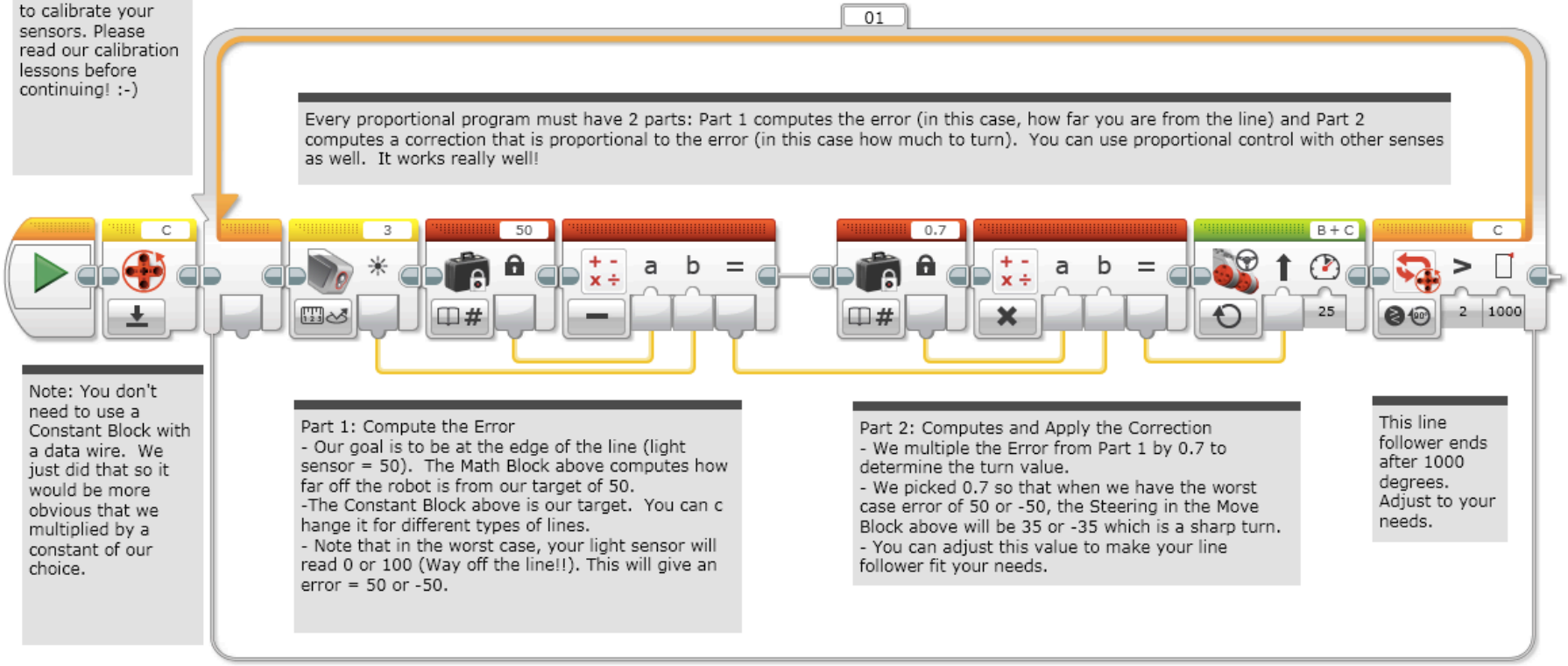


Solution: Proportional Line Follower

Note: This program uses the Color Sensors in Light Mode. This means that you will have to calibrate your sensors. Please read our calibration lessons before continuing! :-)

We recommend that your team uses a proportional line follower like this one. It will be smoothest of the 4 line followers in this lesson. There are even better line followers (that use PID control), but a line follower that uses the "P" is a great start.

A proportional line follower changes the angle of the turn based on how far away from the line the robot is.



Every proportional program must have 2 parts: Part 1 computes the error (in this case, how far you are from the line) and Part 2 computes a correction that is proportional to the error (in this case how much to turn). You can use proportional control with other senses as well. It works really well!

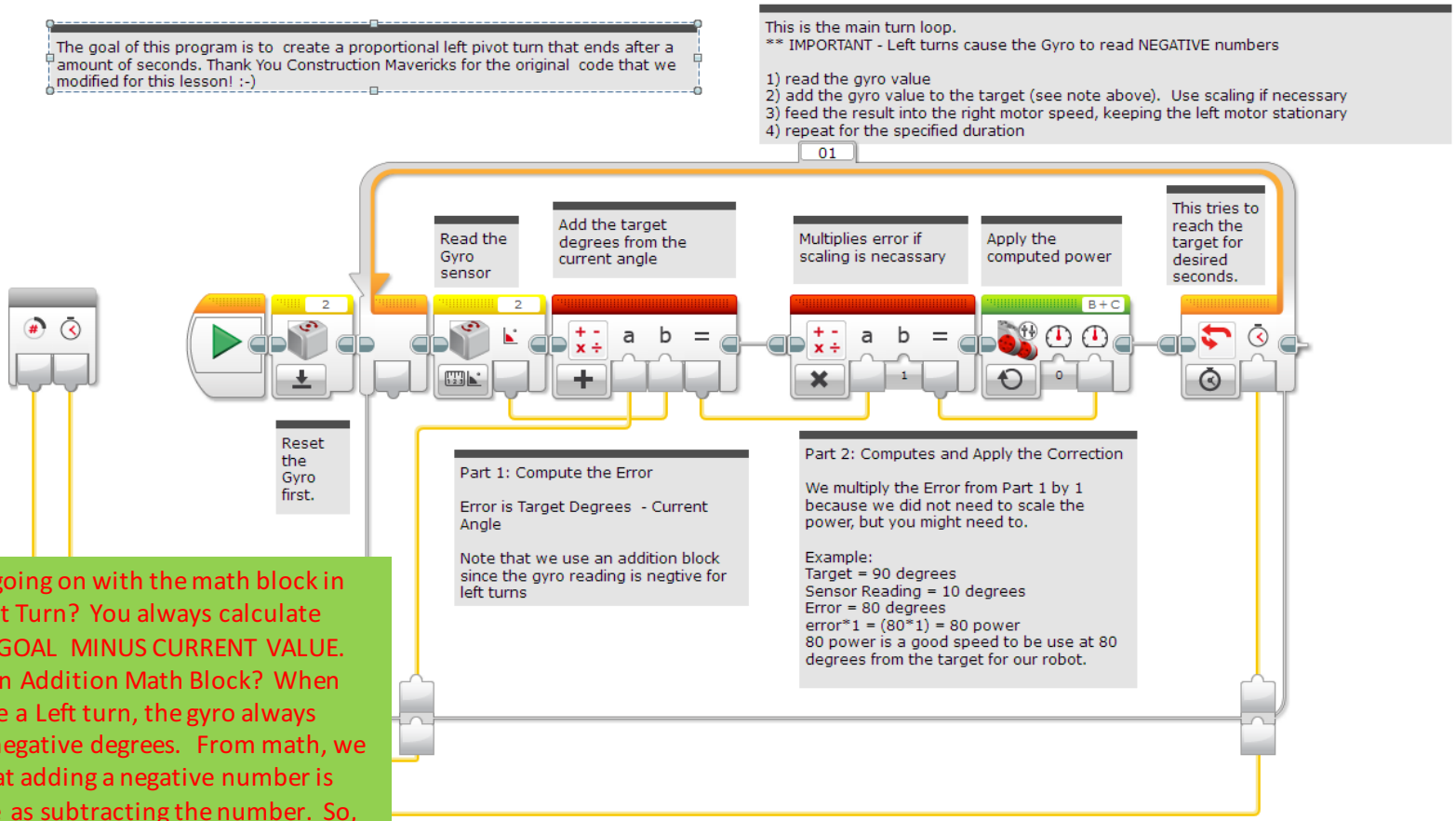
Note: You don't need to use a Constant Block with a data wire. We just did that so it would be more obvious that we multiplied by a constant of our choice.

Part 1: Compute the Error
- Our goal is to be at the edge of the line (light sensor = 50). The Math Block above computes how far off the robot is from our target of 50.
-The Constant Block above is our target. You can c hange it for different types of lines.
- Note that in the worst case, your light sensor will read 0 or 100 (Way off the line!!). This will give an error = 50 or -50.

Part 2: Computes and Apply the Correction
- We multiple the Error from Part 1 by 0.7 to determine the turn value.
- We picked 0.7 so that when we have the worst case error of 50 or -50, the Steering in the Move Block above will be 35 or -35 which is a sharp turn.
- You can adjust this value to make your line follower fit your needs.

This line follower ends after 1000 degrees. Adjust to your needs.

Solution: Gyr0 Left Turn

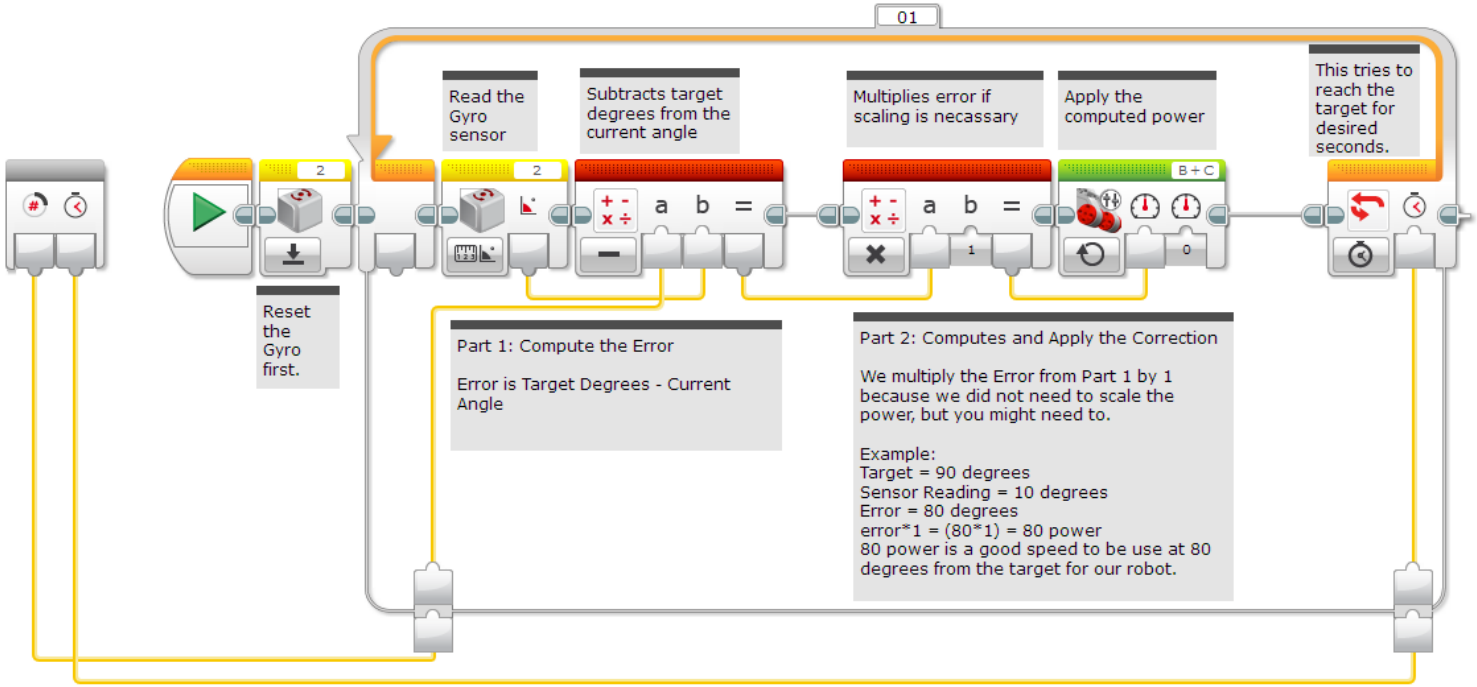


What is going on with the math block in Left Pivot Turn? You always calculate TARGET/GOAL MINUS CURRENT VALUE. So why an Addition Math Block? When you make a Left turn, the gyro always returns negative degrees. From math, we know that adding a negative number is the same as subtracting the number. So, that is why we use the Addition Math block in a Left Gyro Turn.

Solution: Gyro Right Turn

The goal of this program is to create a proportional right pivot turn that ends after a amount of seconds. Thank You Construction Mavericks for the original code that we modified! :-)

This is the main turn loop.
1) read the gyro value
2) subract the gyro value from our target. Use scaling if necessary.
3) feed the result into the left motor speed, keeping the right motor stationary
4) repeat for the specified duration



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 from Droids Robotics (team@droidsrobotics.org).
- Original Gyro Turn code was provided by the Construction Mavericks (frank.levine@gmail.com)
- More lessons at www.ev3lessons.com