ADVANCED PROGRAMMING LESSON

Proportional Control

By Droids Robotics

WHY PROPORTIONAL CONTROL?

- Proportional control is very useful for FLL
- 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 for getting missions done!
- Every proportional control program consists of two stages:
  1. **Computing an error** → how far is the robot from a target
  2. **Making a correction** → make the robot take an action that is proportional to the error (this is why it is called proportional control)

LEARNING WHAT IS PROPORTIONAL

- 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!

LEARN HOW TO CODE PROPORTIONAL CONTROL

To learn how to use proportional control, we give you three different examples:

Dog Follower: uses ultrasonic

- We used proportional ultrasonic moves in Nature's Fury to make sure we hit the Base Isolation Model and the Evacuation Sign just the right amount

Line Follower: uses color sensor

- We use proportional (or full PID) on all lines on the mat to make our moves more efficient

Gyro Turn: uses gyro sensor

- We use proportional control to make sure that we have turned the amount we want

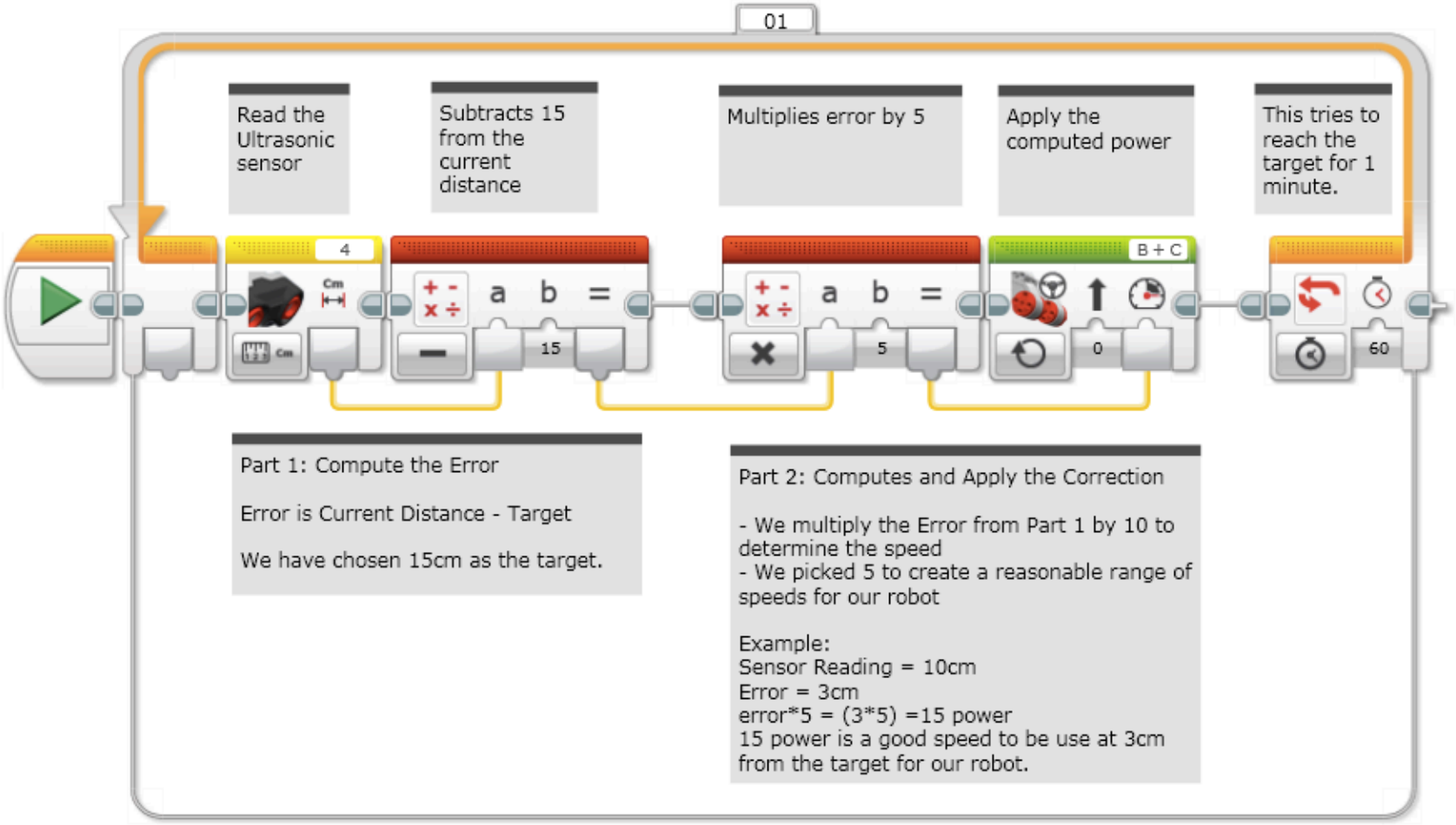
APPLICATIONS OF PROPORTIONAL CONTROL

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

ULTRASONIC: DOG FOLLOWER

We are trying to make a program that stays 7cm away from a moving object. This program uses proportional control.

This code was written by Droids Robotics



Part 1: Compute the Error

Error is Current Distance - Target

We have chosen 15cm as the target.

Part 2: Computes and Apply the Correction

- We multiply the Error from Part 1 by 10 to determine the speed
- We picked 5 to create a reasonable range of speeds for our robot

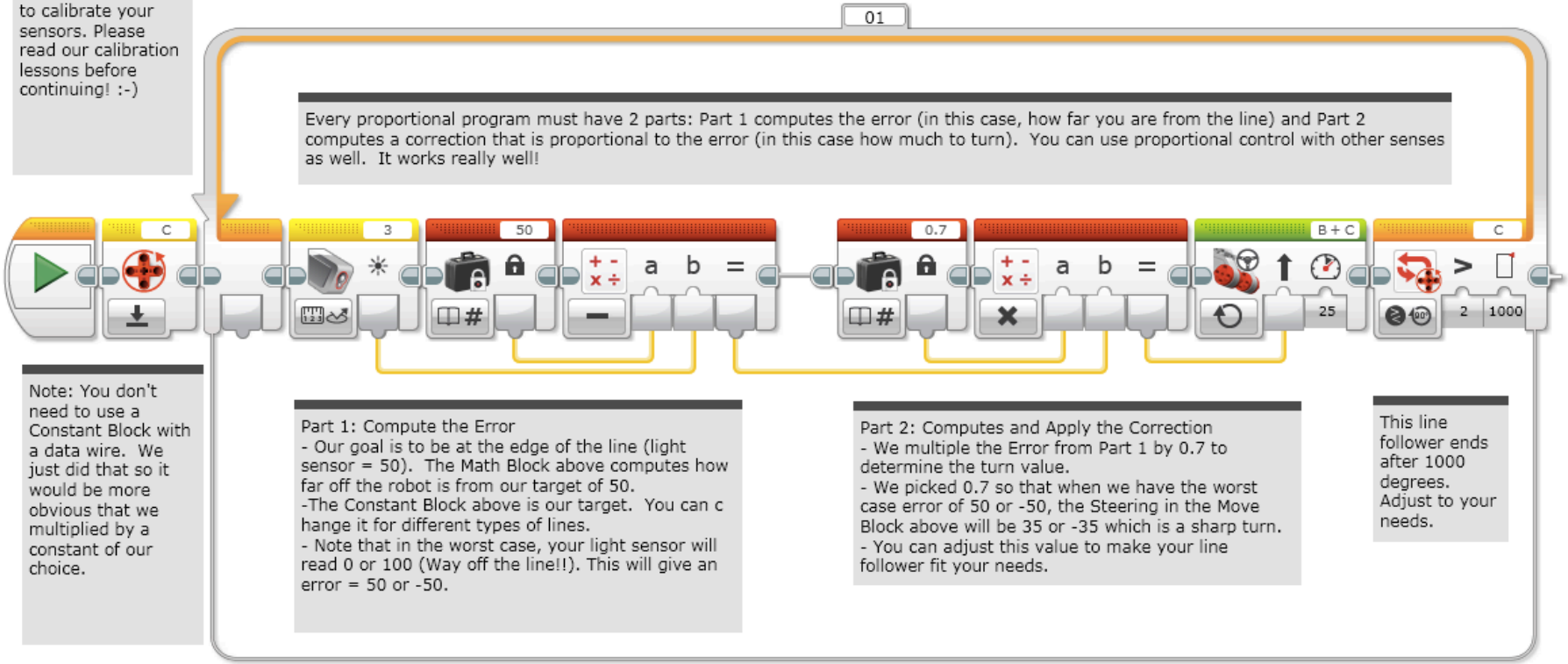
Example:
Sensor Reading = 10cm
Error = 3cm
error*5 = (3*5) =15 power
15 power is a good speed to be use at 3cm from the target for our robot.

COLOR: 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!

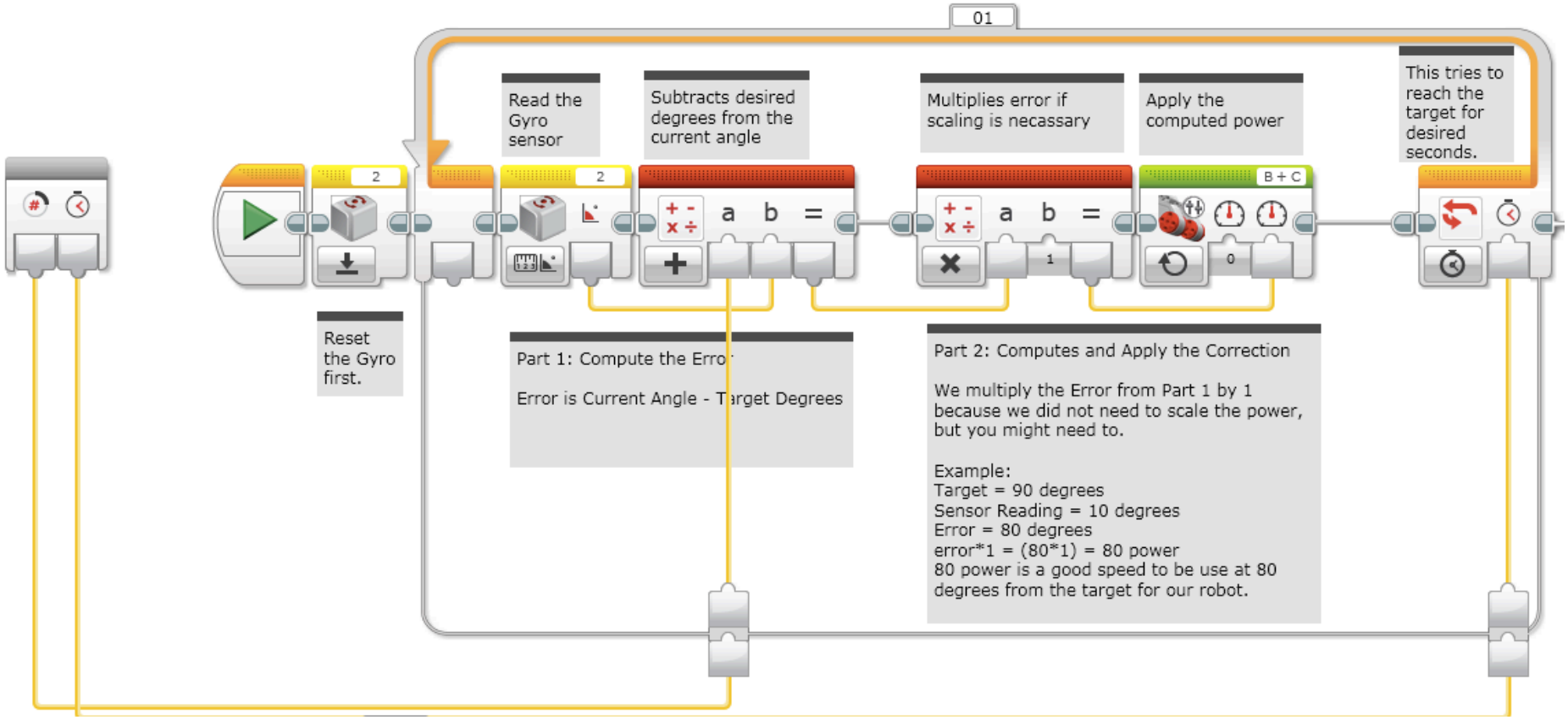
Note: You don't need to use a Constant Block with a data wire. We just did that so it would be more obvious that we multiplied by a constant of our choice.

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.

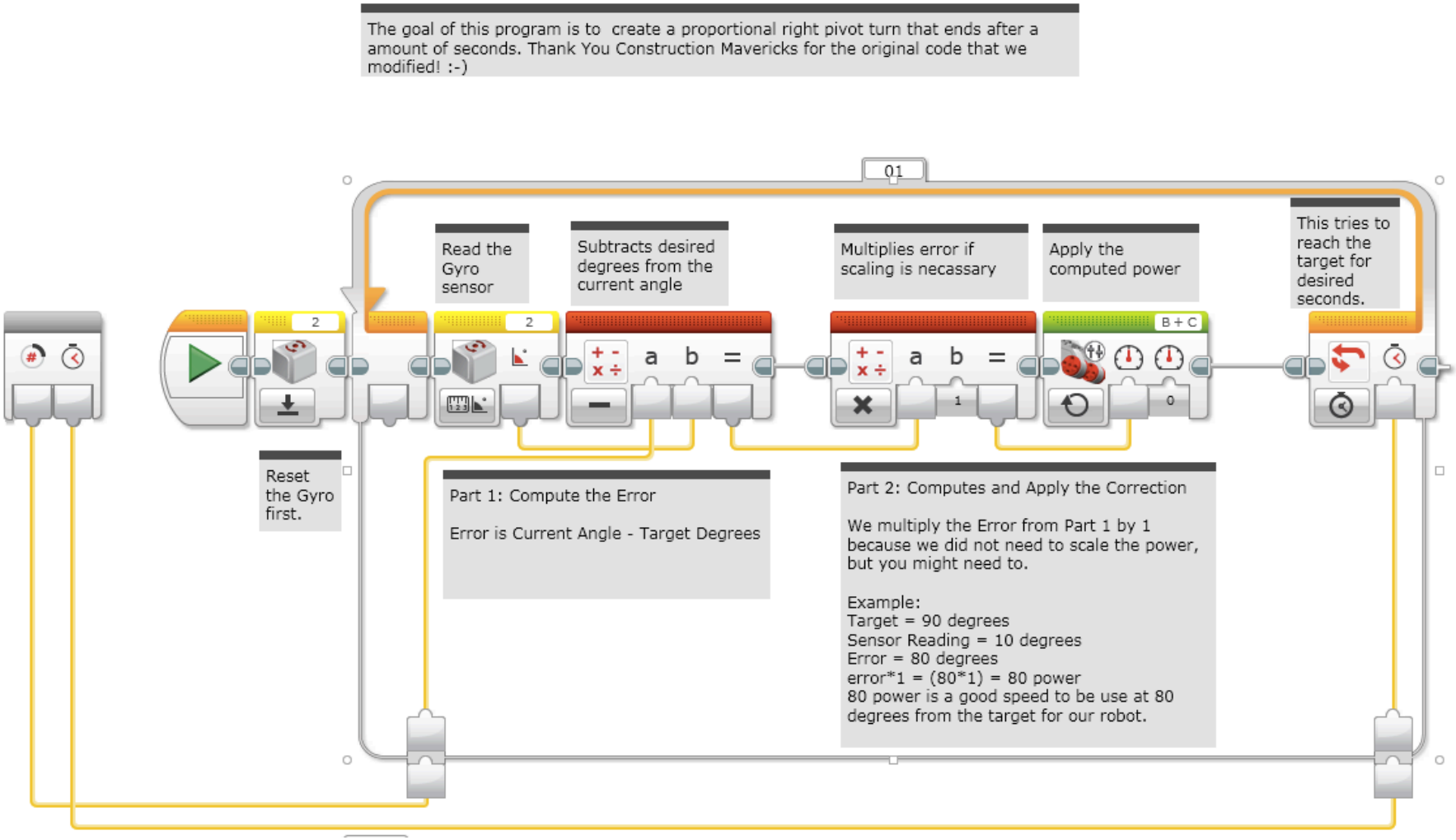
Part 2: Computes and Apply the Correction
- We multiple the Error from Part 1 by 0.7 to determine the turn value.
- We picked 0.7 so that when we have the worst case error of 50 or -50, the Steering in the Move Block above will be 35 or -35 which is a sharp turn.
- You can adjust this value to make your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

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

GYR0: LEFT TURN



GYRO: RIGHT TURN



CREDITS

- This lesson was written by Sanjay Seshan and Arvind Seshan from Droids Robotics
- The original code for the Gyro Turn was from The Construction Mavericks. We modified it a little to use in this lesson.
- The Construction Mavericks can be contacted at: frank.levine@gmail.com
- More lessons at ev3lessons.com
- Contact us: team@droidsrobotics.org