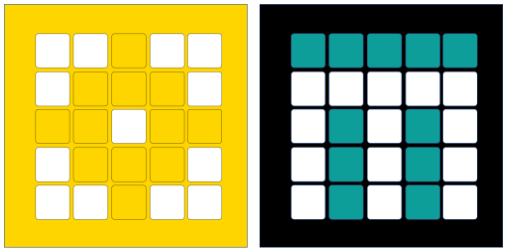
PRIME LESSONS

By the Makers of EV3Lessons



PROPORTIONAL LINE FOLLOWER

BY SANJAY AND ARVIND SESHAN

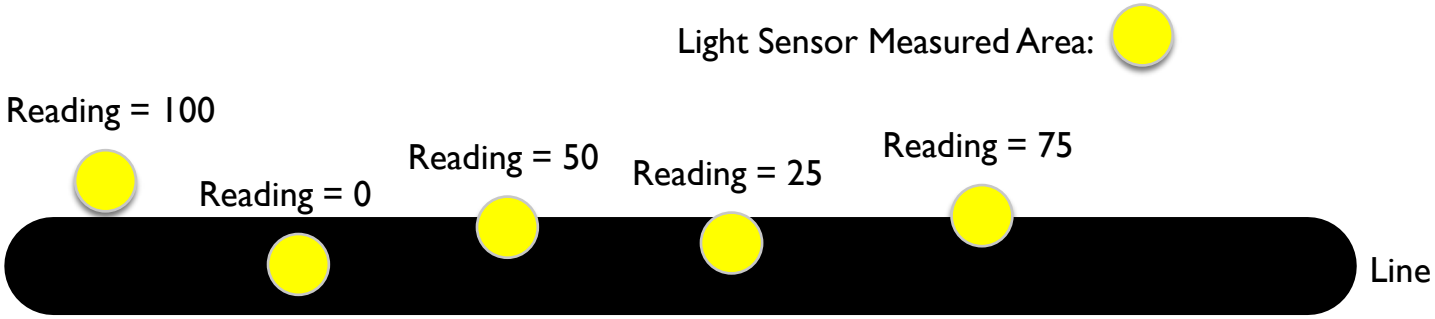
LESSON OBJECTIVES

- Learn to create a proportional line follower
- Learn how to calculate error and correction
- Learn how to use variables and math blocks

© 2020 FLLTutorials, Last edit 05/25/2020

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)



© 2020 FLLTutorials, Last edit 05/25/2020

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

© 2020 FLLTutorials, Last edit 05/25/2020

HOW DO YOU MAKE A PROPORTIONAL LINE FOLLOWER?

Pseudocode:

1. Compute the error = Distance from line = (Light sensor reading - Target Reading)
2. Scale the error to determine a correction amount. Adjust your scaling factor to make you robot follow the line more smoothly.
3. Use the Correction value (computed in Step 2) to adjust the robot's turn towards the line.

© 2020 FLLTutorials, Last edit 05/25/2020

CHALLENGE

Compute Error	error
Distance from line = (Light sensor reading - Target Reading)	`error = color.get_reflected_light() - 50`
Compute Correction	`correction = error * 0.3`
Scale the error to determine a correction amount. Use this to adjust power input on move block.	

Apply Correction	`motor_pair.start_tank(int(40+correction), int(40-correction))`
	`# Note that the int function is used on the inputs to ensure that`
	`# speed is an integer`

Use the correction and a base power to control each motor.	

© 2020 FLLTutorials, Last edit 05/25/2020

PROPORTIONAL LINE FOLLOWER

```
from spike import PrimeHub, LightMatrix, Button, StatusLight, ForceSensor, MotionSensor, Speaker, Colo
rSensor, App, DistanceSensor, Motor, MotorPair
from spike.control import wait_for_seconds, wait_until, Timer
from math import *

hub = PrimeHub()

color = ColorSensor('F')
motor_pair = MotorPair('A', 'E')

while True:
    error = color.get_reflected_light() - 50
    correction = error * 0.3

    motor_pair.start_tank(int(40+correction), int(40-correction))
    # Note that the int function is used on the inputs to ensure that
    # speed is an integer
```

Part 1: Compute the Error
Our goal is to stay at the edge of the line (light sensor = 50)

Part 2: Apply the correction
The error in part 1 is multiplied by a Constant of Proportionality (0.3). This will be different for each robot/application. See slide 8 to learn how to tune this number.

© 2020 FLLTutorials, Last edit 05/25/2020

KEY STEP: TUNING THE CONSTANT

- Note, the 0.3 constant in the previous slide is specific to our robot – you need to tune this value for yourself
- This constant is called the Proportional Constant, or Constant of Proportionality

- The most common way to tune your constant is trial and error.
- This can take time. Here are some tips:
  - Start with your constant as1.0 adjust by ±0.5 initially
  - Adjust to a point where the controller is pretty smooth
  - Adjust ±0.1 for fine tuning

© 2020 FLLTutorials, Last edit 05/25/2020

CREDITS

- This lesson was created by Sanjay Seshan and Arvind Seshan for SPIKE Prime Lessons
- More lessons are available at www.primelessons.org

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.