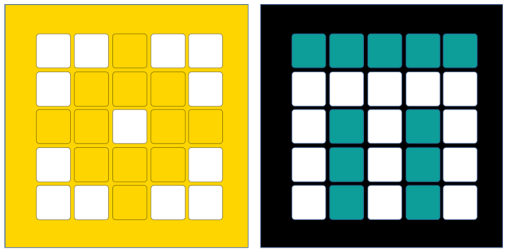
PRIME LESSONS

By the Makers of EV3Lessons



PROPORTIONAL LINE FOLLOWER

BY SANJAY AND ARVIND SESHAN

This lesson uses SPIKE 3 software

LESSON OBJECTIVES

Learn to create a proportional line follower

Learn how to calculate error and correction

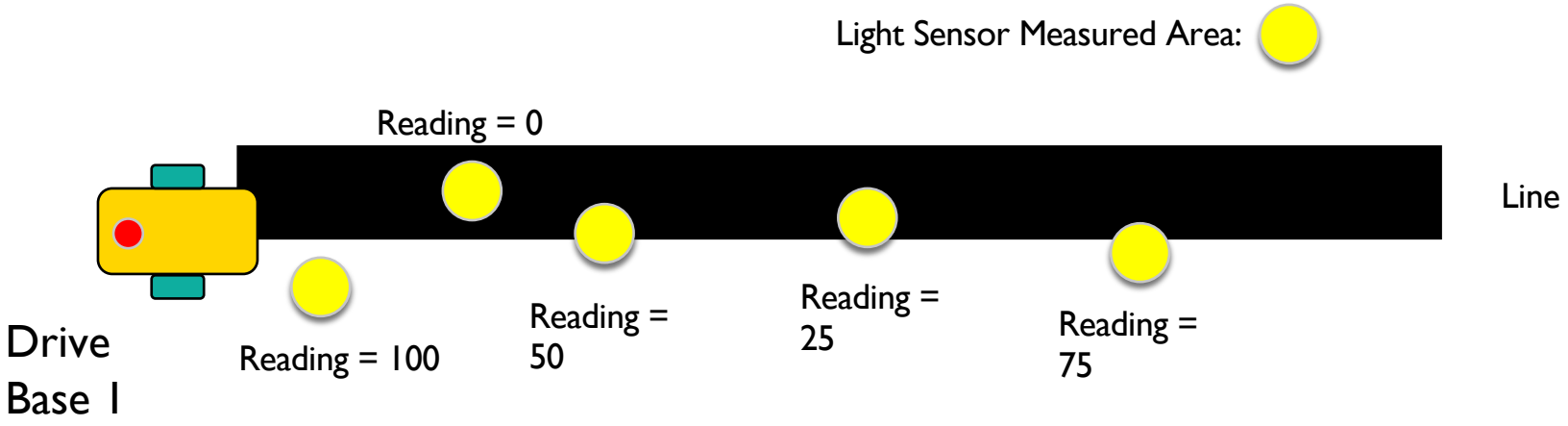
Learn how to use variables and math blocks

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)

For following the right side of a black line (i.e. Black-White edge):



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

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.

CHALLENGE

For following a Black-White edge:

Compute Error	Error
Distance from line = (Light sensor reading - Target Reading)	`error = 50 - color_sensor.reflection(port)` The maximum absolute value of error is 50
Compute Correction	`correction = int(error * 0.5)`
Scale the error to determine a correction amount. Use this to adjust power input on move block.	The int function converts the result to an integer to use in the move function. Since the maximum absolute value of error is 50, the correction ranges from -25 to 25
Apply Correction	`motor_pair.move(motor_pair.PAIR_1, correction, velocity = 300)`
Use the correction to steer the motor pair. You can also use it (scaled appropriately) to adjust the base velocity of each motor, if using tank mode.	

PROPORTIONAL LINE FOLLOWER

```
from hub import port
import motor, motor_pair, color_sensor, runloop, sys

# Constants for Drive Base 1
motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

# Follow the right side of black line (Black-White edge).
# To follow a White-Black edge, change the error condition to (reflection - 50)
async def line_follow_forever():
    while (True):
        # Compute the error
        error = 50 - color_sensor.reflection(port.A)
        # Compute the correction by multiplying the error
        # by a Constant of Proportionality
        correction = int(error * 0.5)
        motor_pair.move(motor_pair.PAIR_1, correction, velocity = 300)

async def main():
    await line_follow_forever()

runloop.run(main())
```

KEY STEP: TUNING THE CONSTANT

Note, the 0.5 constant in the previous slide is specific to our robot (Drive Base 1 design). It is a good start, but you may need to tune this value for your specific robot.

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 a value of 0.5 and adjust ±0.05 for fine tuning
- If you are using the steering method, try to keep the correction value from -30 to 30.
- Adjust to a point where the controller is pretty smooth

CHALLENGE

Convert the example programs in the Basic line follower lesson to use a proportional line follower:

- Line follow until second sensor sees black
- Line follow for approximate distance

CREDITS

This lesson was created by Sanjay Seshan and Arvind Seshan for SPIKE Prime Lessons

Additional contributions by FLL Share & Learn community members.

More lessons are available at www.primelessons.org