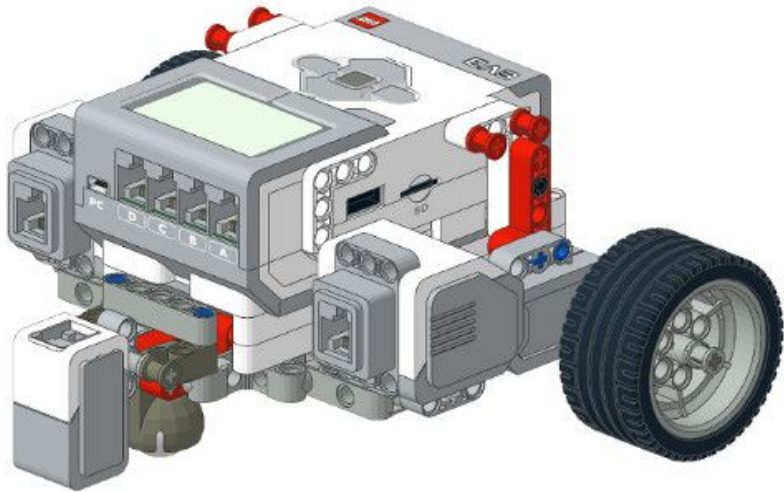




**MINDSTORMS**  
EV3

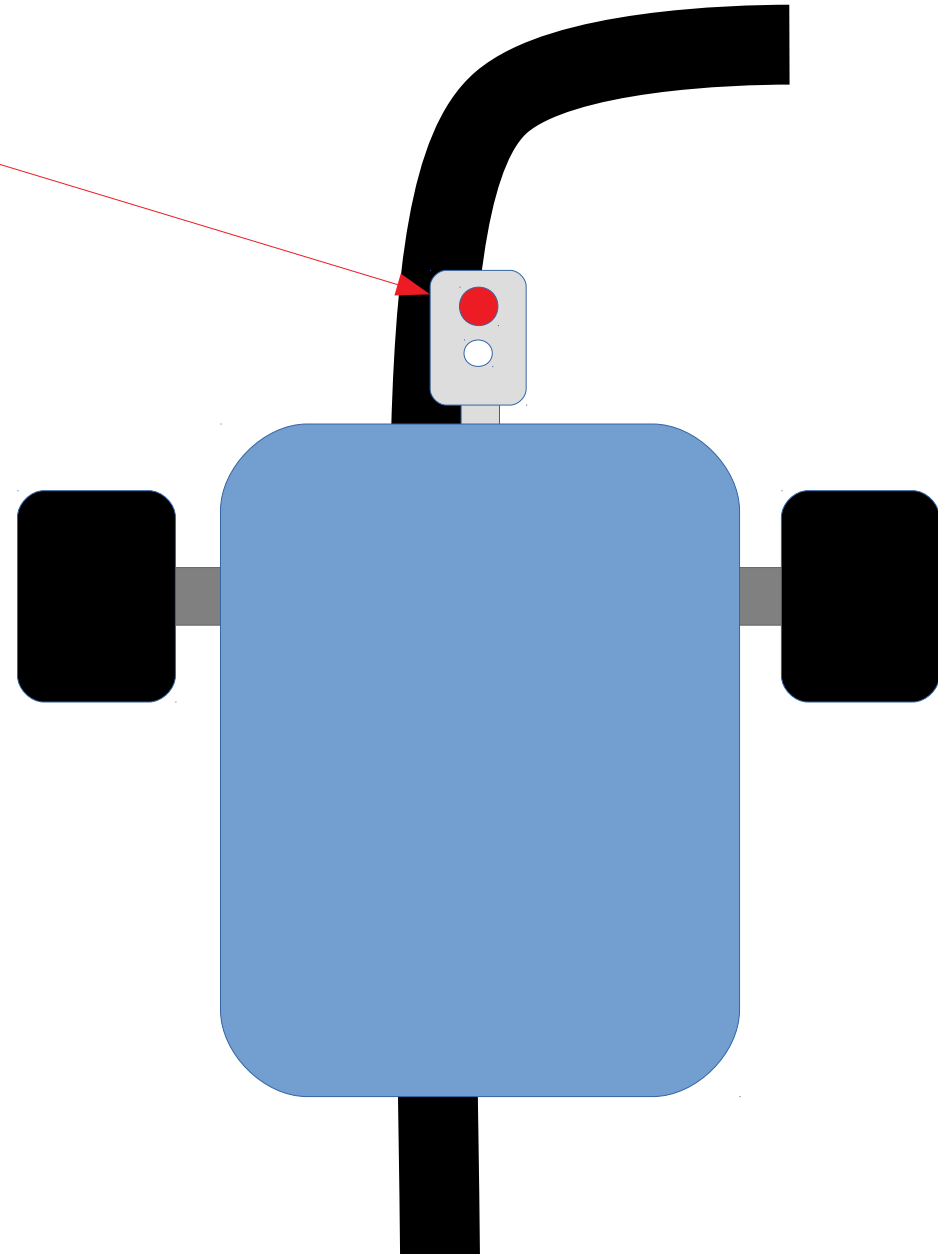
# SINGLE SENSOR LINE FOLLOWER



**A POSTERIORI**  
Play · Experience · Learn

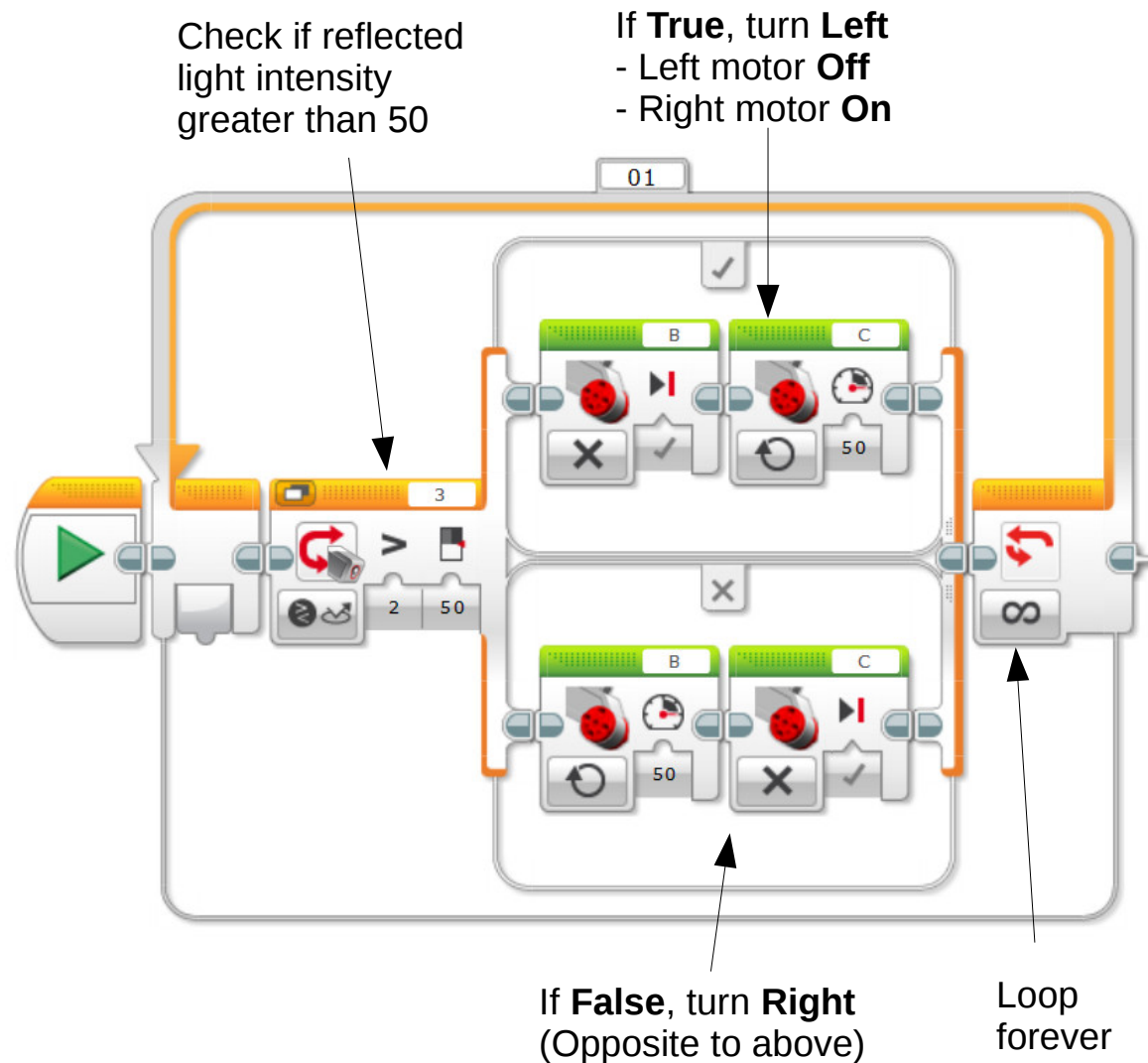
# One Sensor Line Following

- Sensor on edge of line
- If sensor is reading...
  - White: Robot is too far right and needs to turn left
  - Black: Robot is too far left and needs to turn right



# 2 States Algorithm

- Loops forever
- Switch monitors reflected light
  - White ( $>50$ ): Turn Left
  - Black ( $<50$ ): Turn Right
- Robot “wiggles” left and right

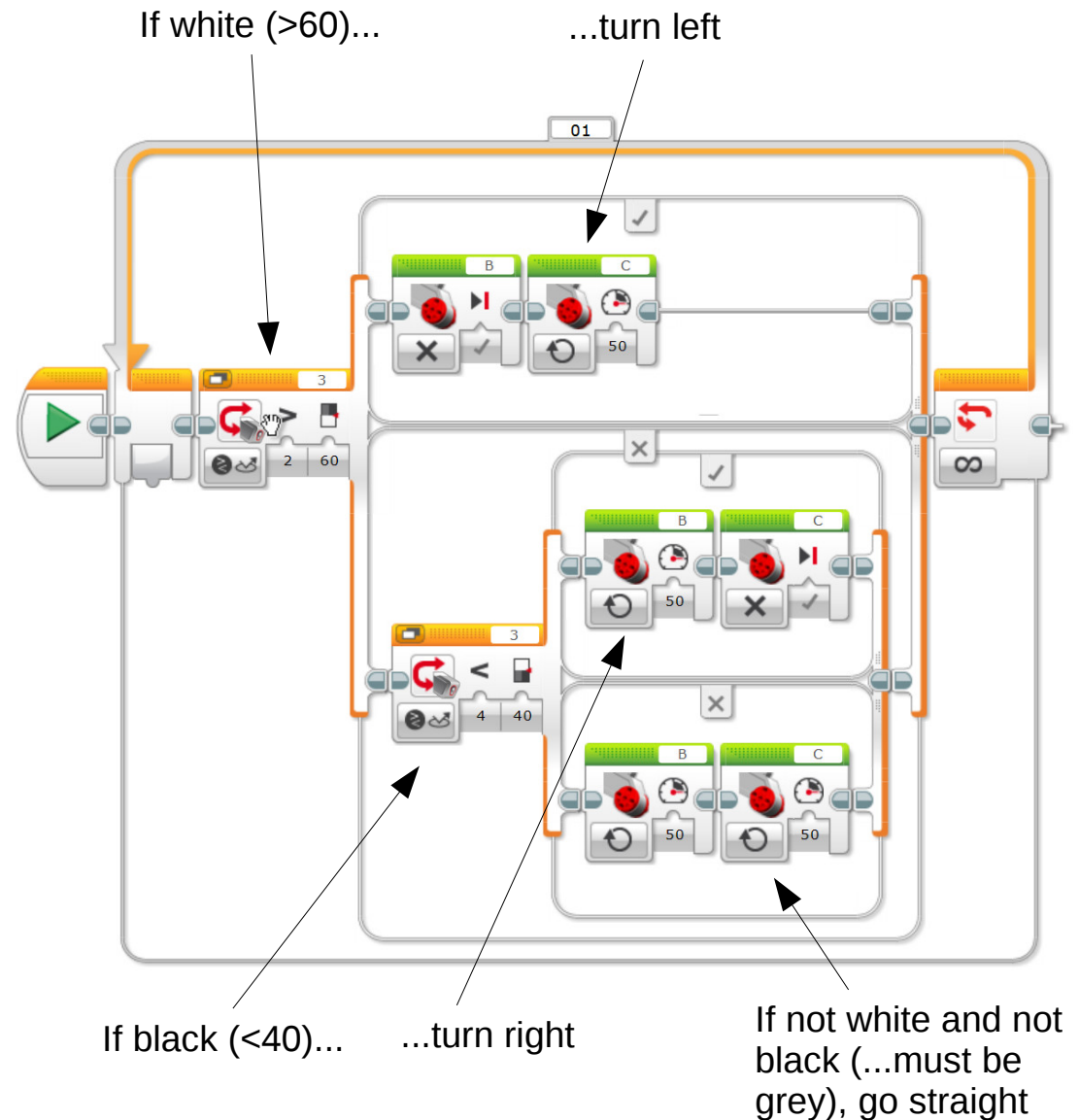


# Common Problems

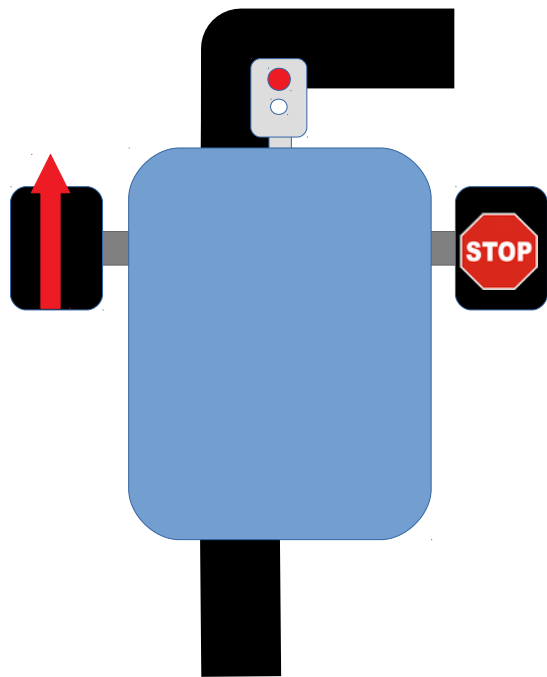
- Movement is slow and jerky
- Robot overshoots curve
  - EV3 takes some time to read light sensor and adjust motor speed
  - If moving too fast, robot may overshoot the line before it has time to correct
  - Go slower!
- Robot works well on smooth lines but fails on sharp bends

# 3 States Algorithm

- Check for **Black**, **White**, and **Grey**
  - White ( $>60$ ): Turn Left
  - Black ( $<40$ ): Turn Right
  - Grey (Between 40 to 60): Go Straight
- Robot runs smoother, but still can't handle sharp turns

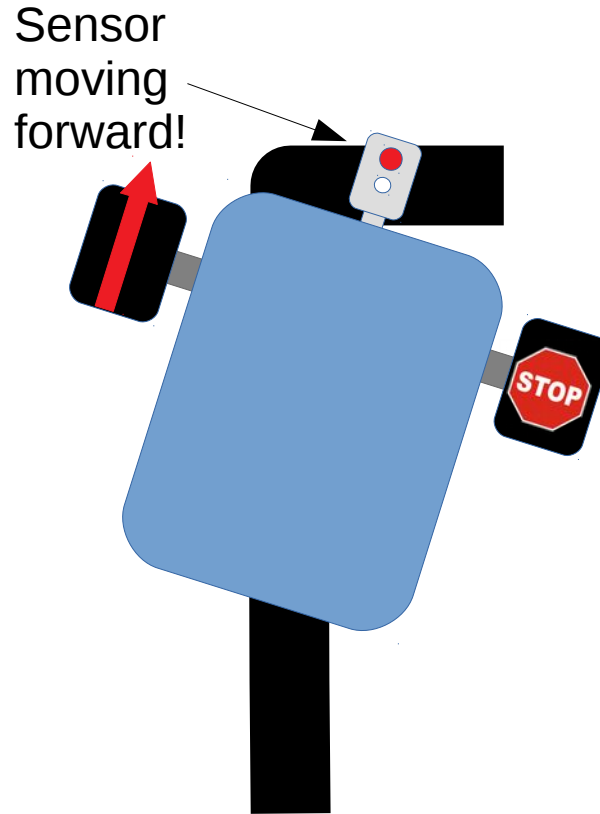


# Problem with sharp turns



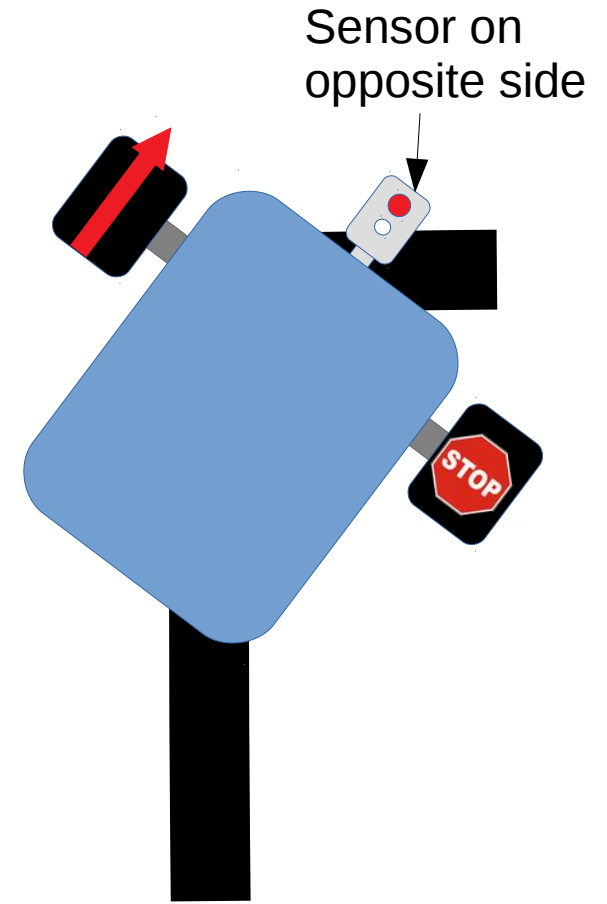
Sensor detects black

- Robot turns Right
- Right motor Stop
- Left motor Forward



Sensor still on black

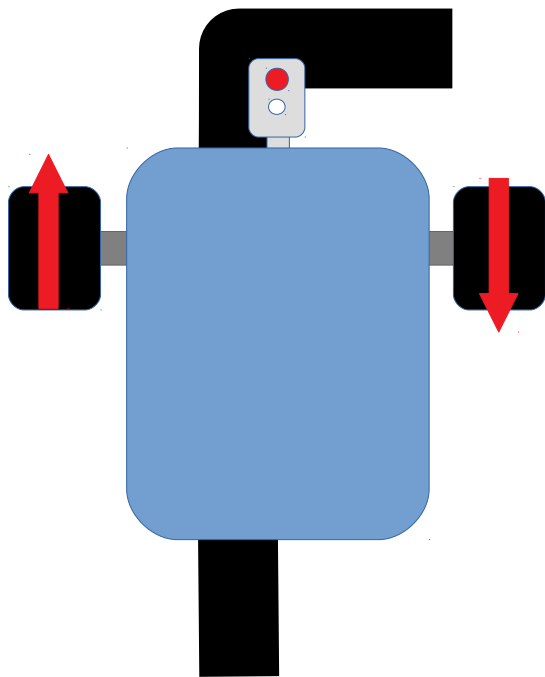
- Robot continue turning
- Sensor moving forward!



Sensor on opposite side of line!

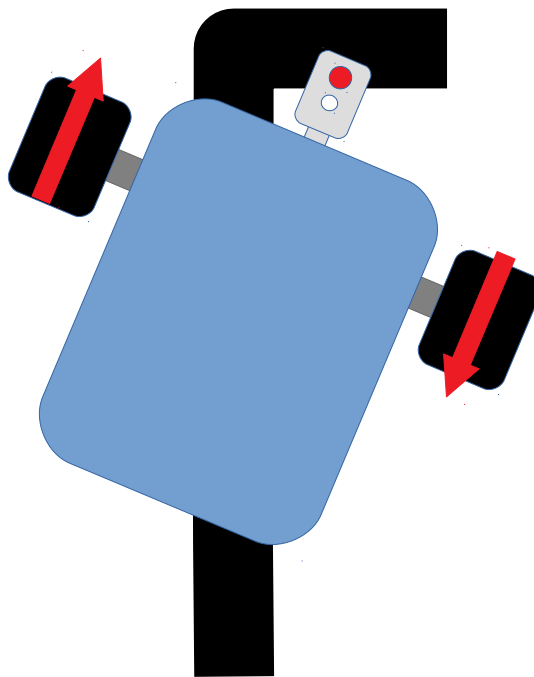
- Robot sees white
- Turns Left (wrong way)
- Fail!

# Handling sharp turns



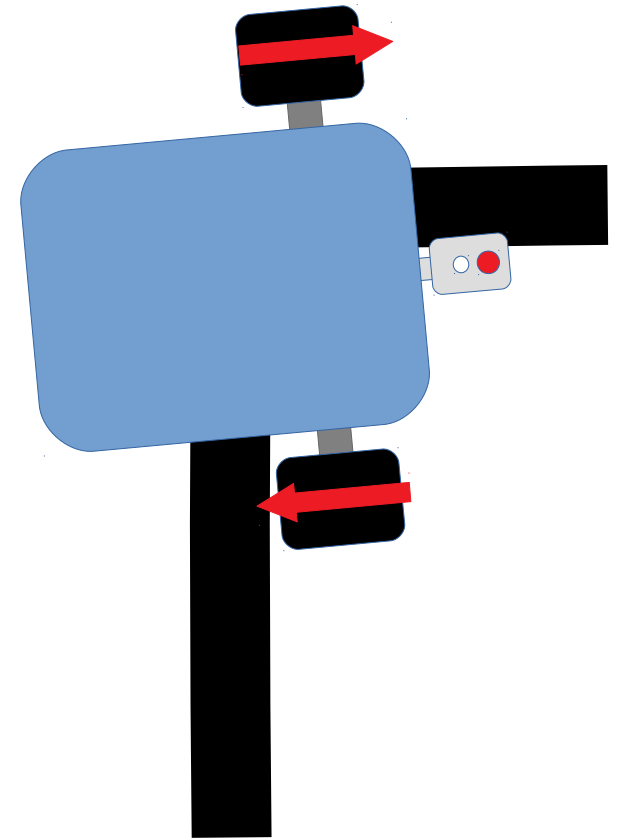
Sensor detects black

Robot turns Right  
- Right motor **Reverse**  
- Left motor forward



Sensor still on black

Robot turns but sensor **NOT** moving forward




Robot completes turn

Continue with line following


# 5 States Algorithm


- Checks for...

- Very White ( $>90$ ): Turn Hard Left 

- Slightly White (60 to 90): Turn Left 

- Grey (40 to 60): Go Straight 

- Slightly Black (10 to 40): Turn Right 

- Very Black ( $<10$ ): Turn Hard Right 

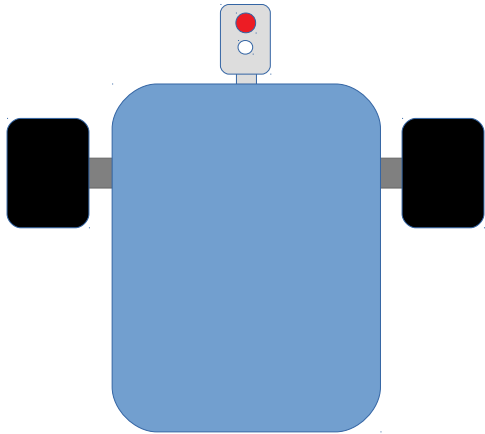
- **Figure out the program yourself!**



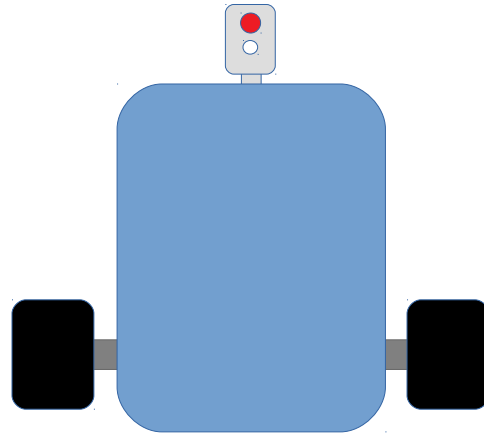
# Tips and Tricks

- Calibrate sensors for consistent result
- Use “Port View” and tests to figure out the best values to use for sensor and motor
  - The best values are different for every robot!
- When making a hard turn, the reverse motor **need not** be the same magnitude as the forward (ie. You can have Forward 40 and Reverse -20)
- Going slow can solve some problems and make it easier to debug others

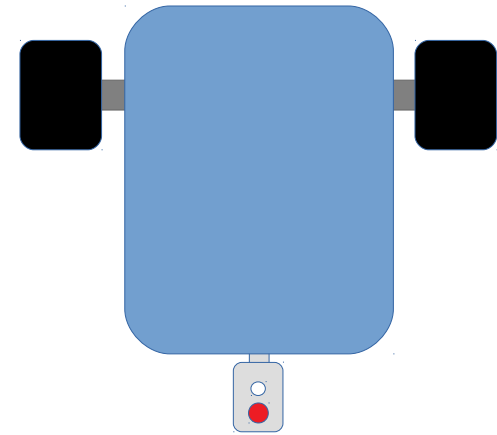
# Things to Think About



Sensor in-front of  
wheels





Sensor far in-front of  
wheels



Sensor behind  
wheels

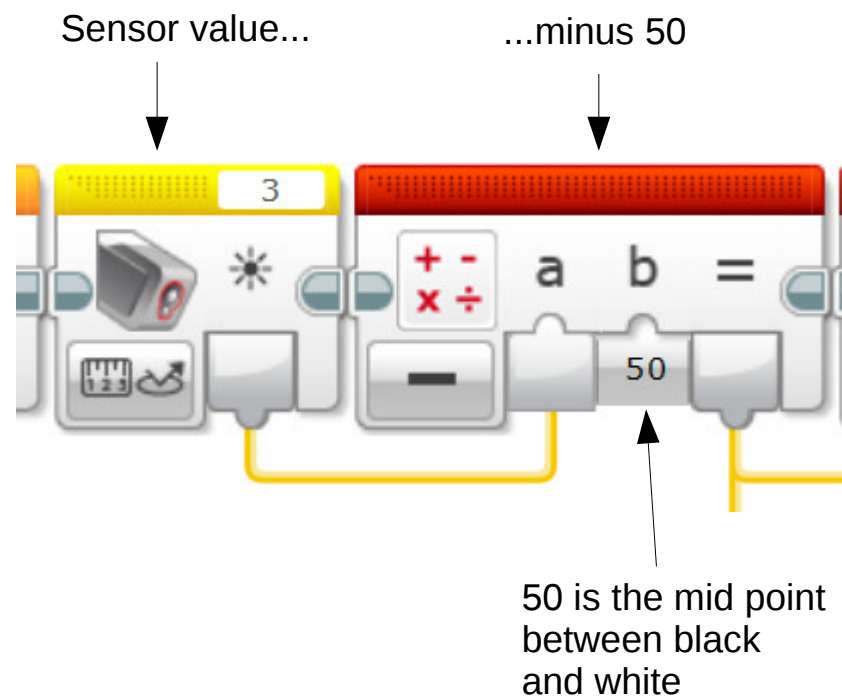
**How does the sensor position affects the robot?**

# Proportional Controls (Adv)

- We have looked at 2 States, 3 States, and 5 States
- In 5 States Algorithm
  - Slightly black: Turn slightly right 
  - Very black: Turn hard right 
- Proportional Control = Infinitely Many States
  - The whiter it is, the more you turn left
  - The blacker it is, the more you turn right
  - **Exactly** 50% grey, go straight

# Proportional Control (Adv)

- Take “sensor value minus 50” we’ll call this the **Error**
- When the error is...
  - Positive: White
    - 1 : Very slightly white
    - 2 : More white
    - 3 : Even more white
    - 4, 5, 6, etc... until 50 (max)
  - Negative: Black
    - -1 : Very slightly black
    - -2, -3, -4, etc... until -50 (min)
  - Exactly Zero: 50% Grey
    - Sensor is in the middle of the line
    - No error



# Proportional Control (Adv)

- Use the **Error** to adjust motor **Base Speed**

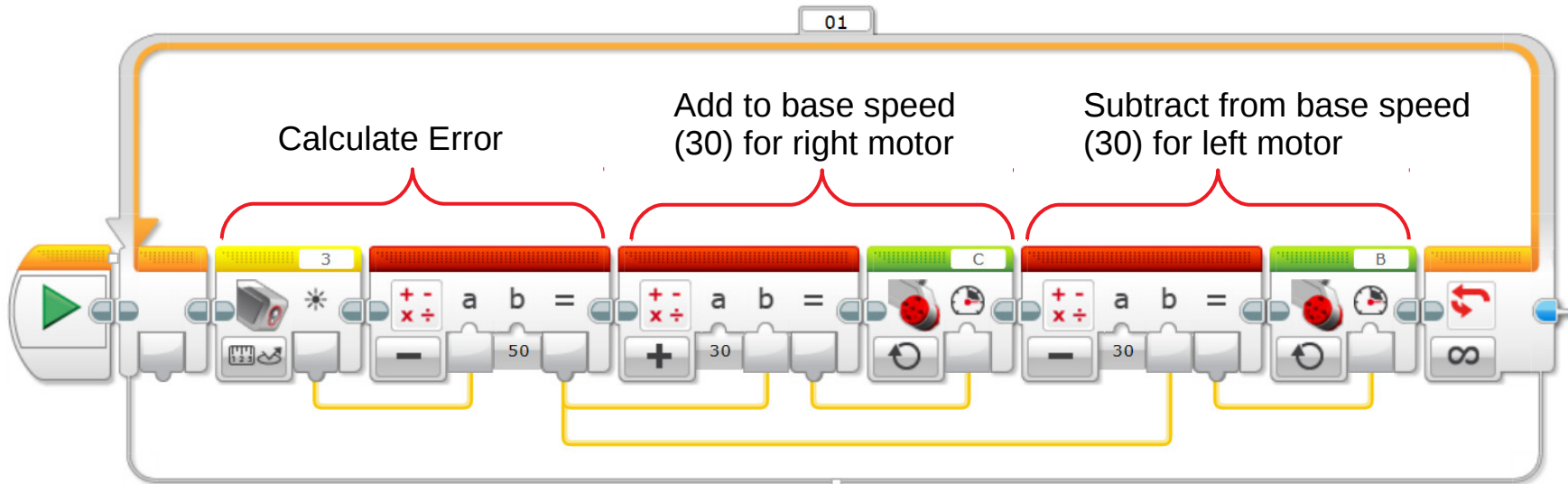
- If white (correction is **positive**)...

- We want to turn **left**
- Right motor needs to go faster; **Add** error
- Left motor needs to go slower; **Subtract** error

- If black (correction is **negative**)...

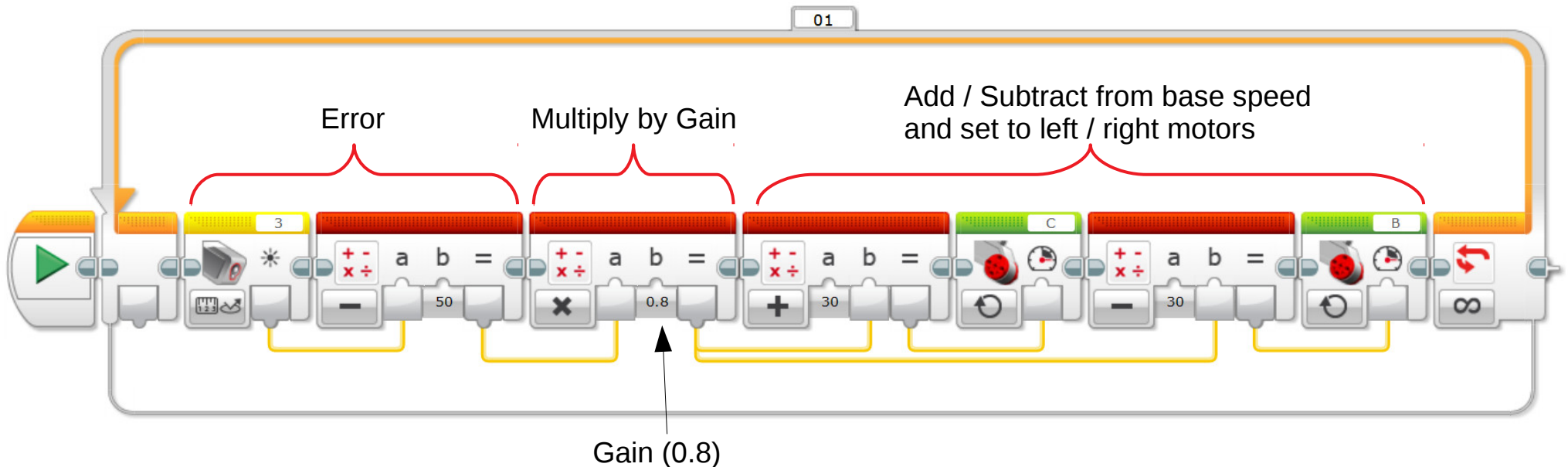
- We want to turn **right**
- Right motor needs to go slower; **Add** error
- Left motor needs to go faster; **Subtract** error

Same operations!

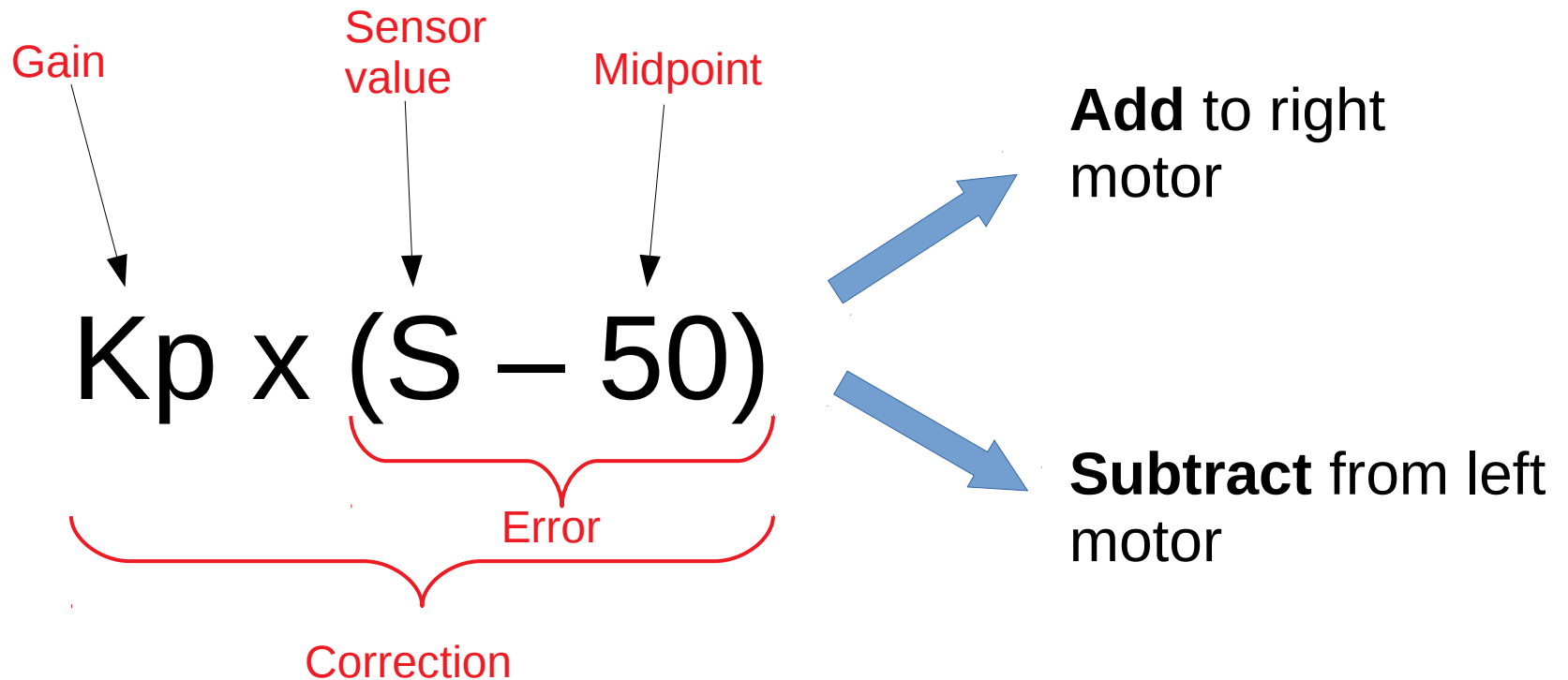


# Proportional Control (Adv)

- Multiplying by a **Gain** lets you adjust how strongly the robot corrects
- Without gain, you cannot tune your robot
- Test to find out the best gain for your robot (suggested testing values from 0.2 to 1.0)



# Proportional Control (Adv)

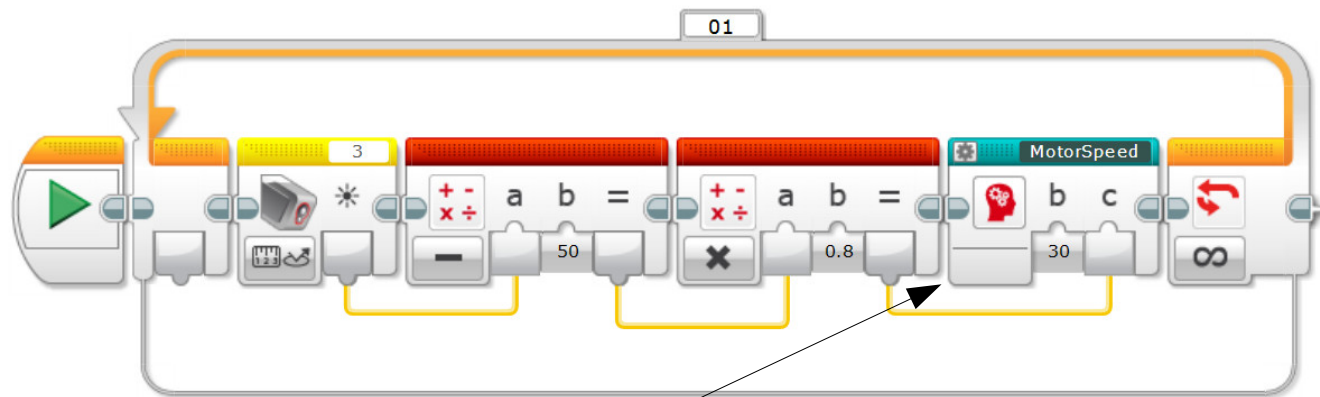


These are standard engineering terminology. Professional engineers use these terms to make themselves sound smarter. You should do the same!

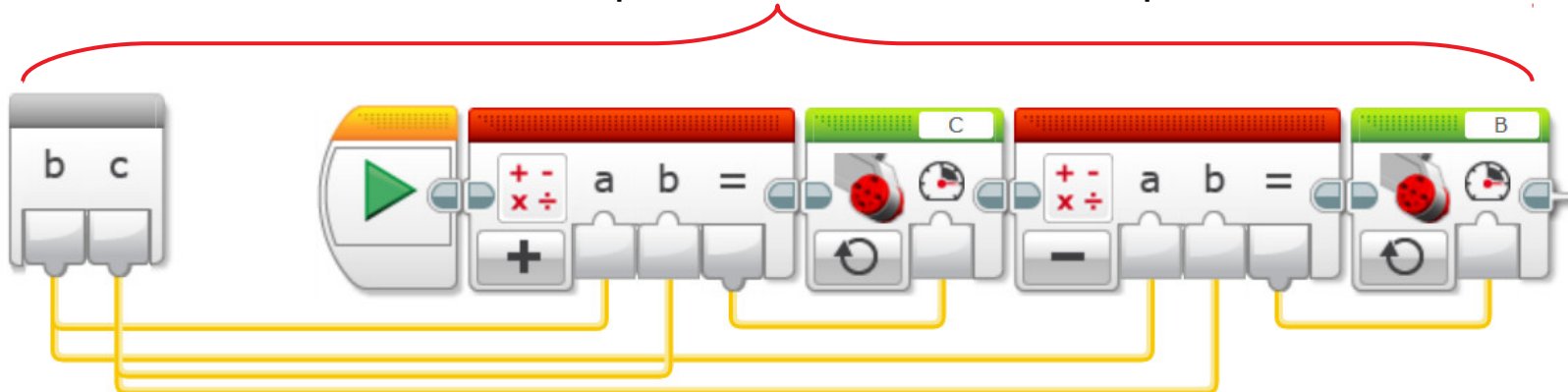
\* The “p” in “Kp” stands for proportional. In a full PID (Proportional, Integral, Derivative) control, you will also have an “Ki” and “Kd”.

# Tips and Tricks

- Use **My Blocks** to keep your program well organized



“MotorSpeed” My Blocks, takes a base speed and correction as inputs





# Copyright

- Created by A Posteriori LLP
- Visit <http://aposteriori.com.sg/> for more tips and tutorials
- This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.



**A POSTERIORI**

Play · Experience · Learn