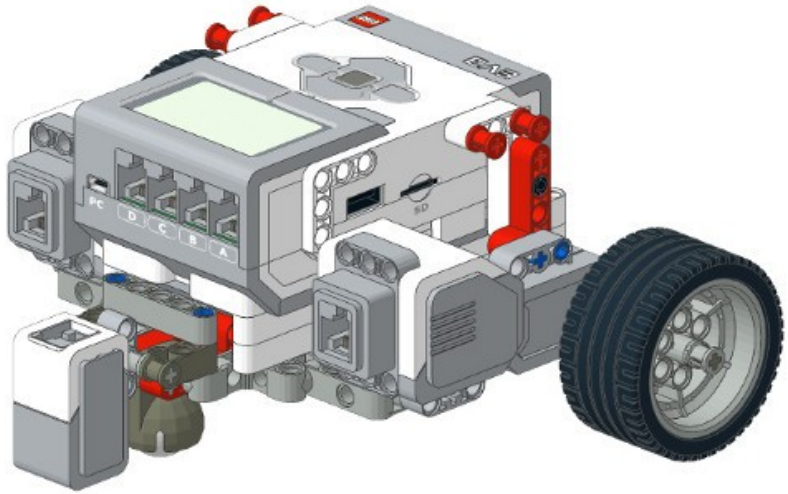
A POSTERIORI

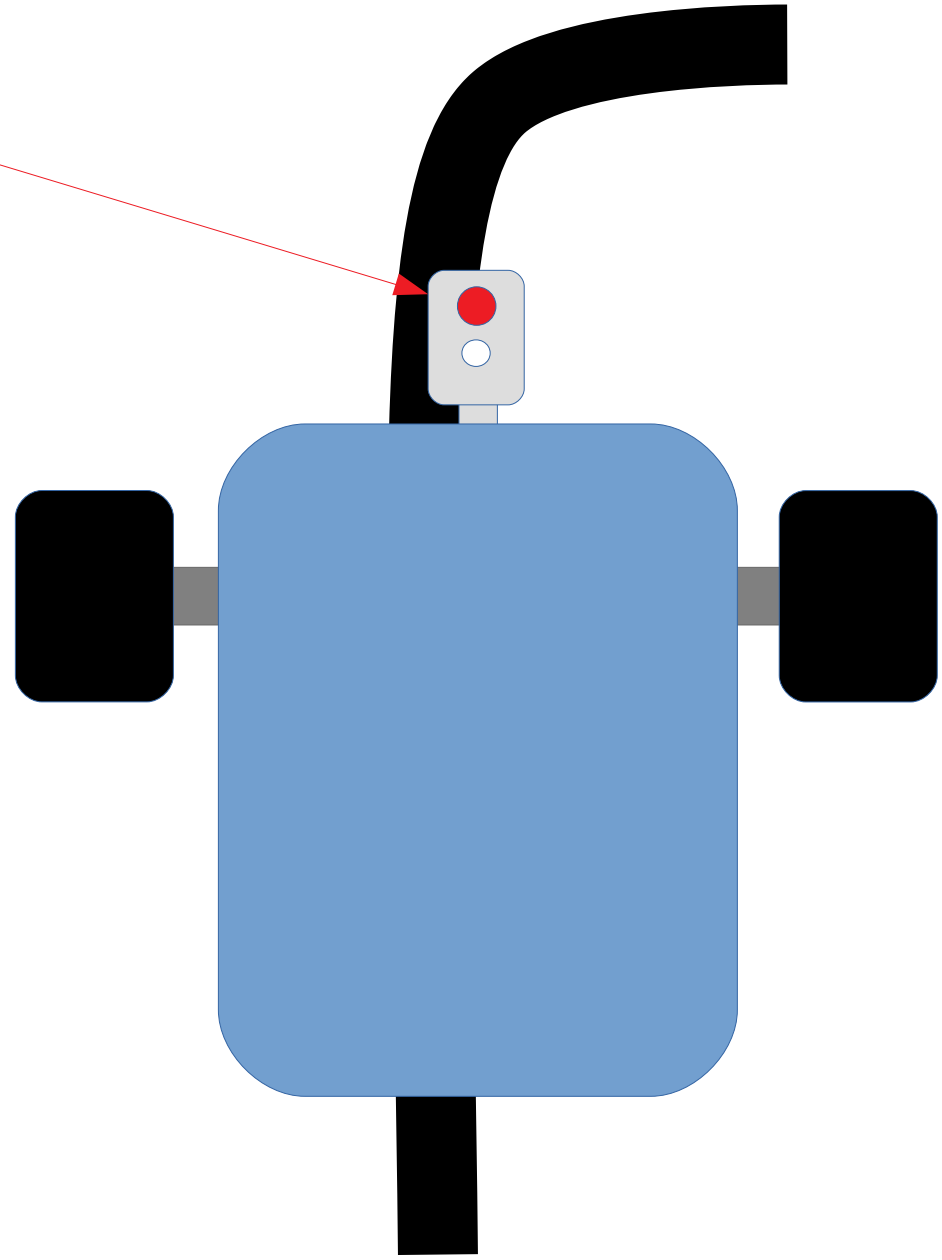
Play · Experience · Learn

SINGLE SENSOR LINE FOLLOWER



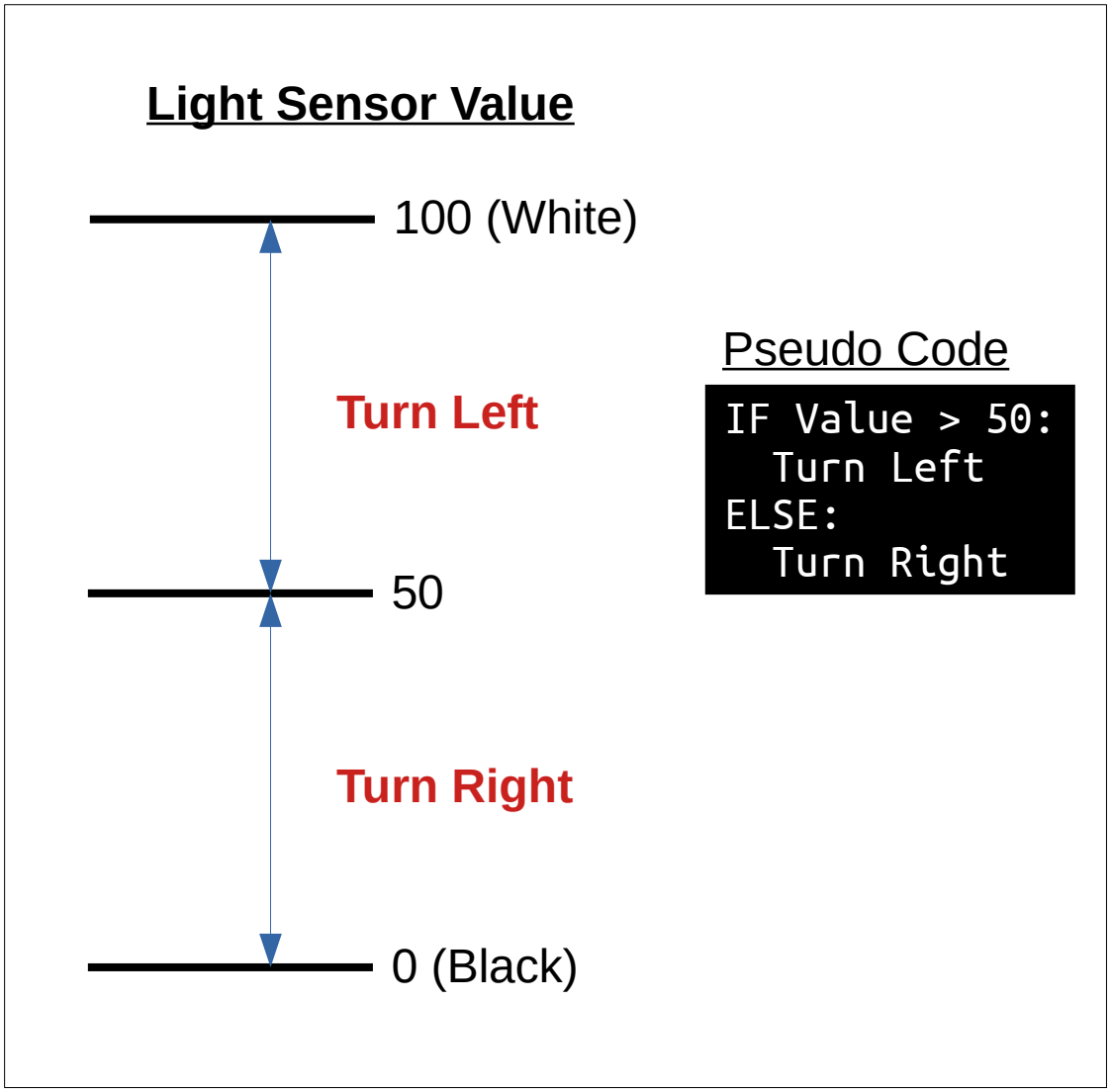
One Sensor Line Following

- 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

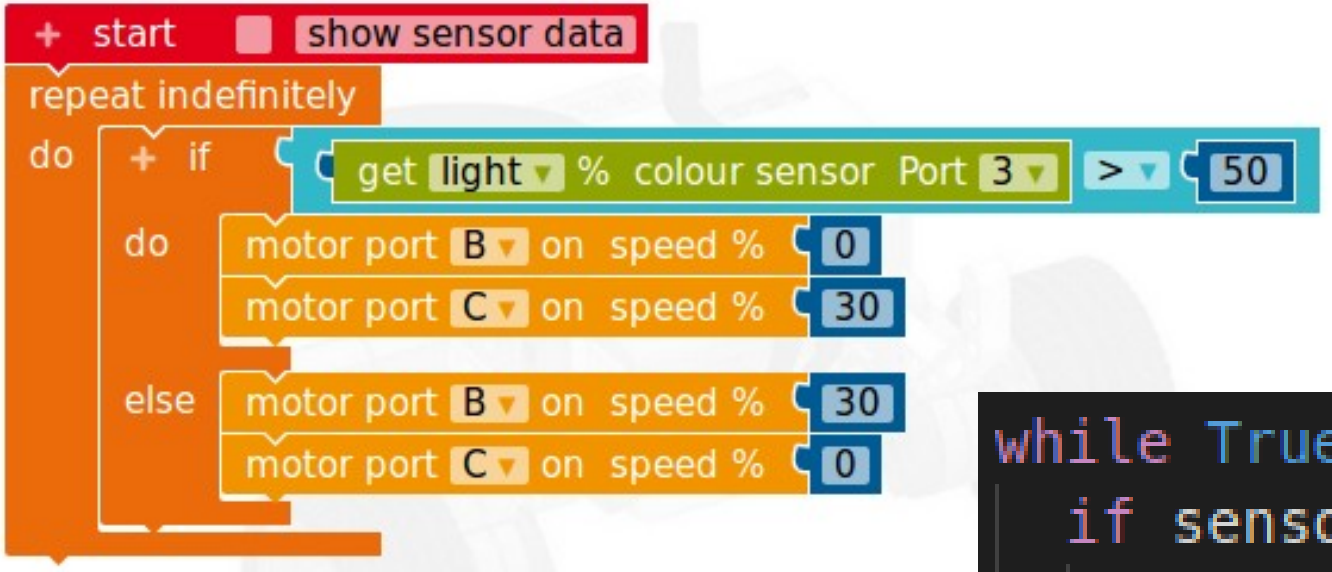


2 States Algorithm

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



2 States Algorithm




```
while True:
    if sensor.value > 50:
        lMotor.speed_sp = 0
        rMotor.speed_sp = 300
    else:
        lMotor.speed_sp = 300
        rMotor.speed_sp = 0
    lMotor.run_forever()
    rMotor.run_forever()
```

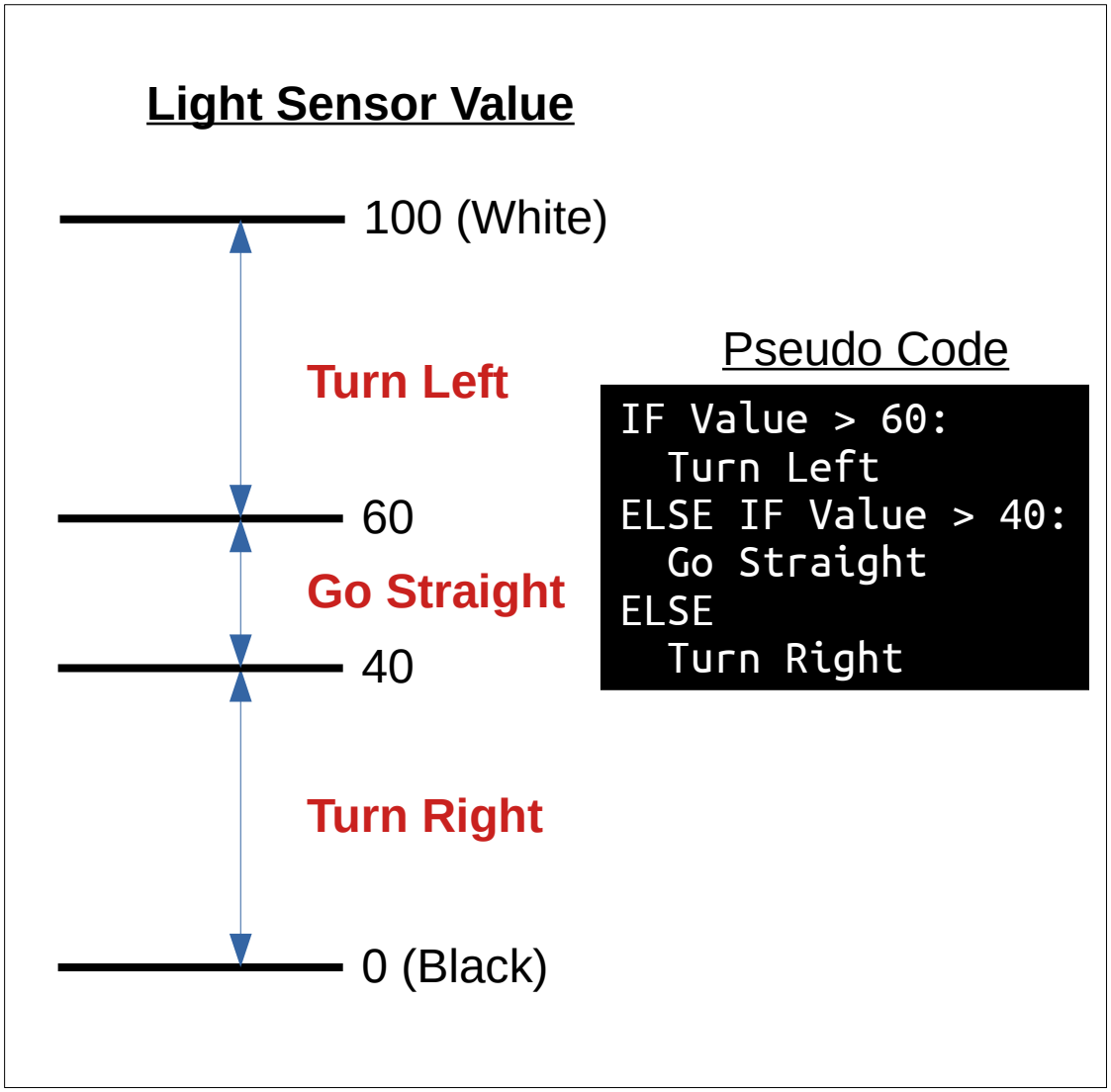
Initialization of motors and sensor not shown here.

Common Problems

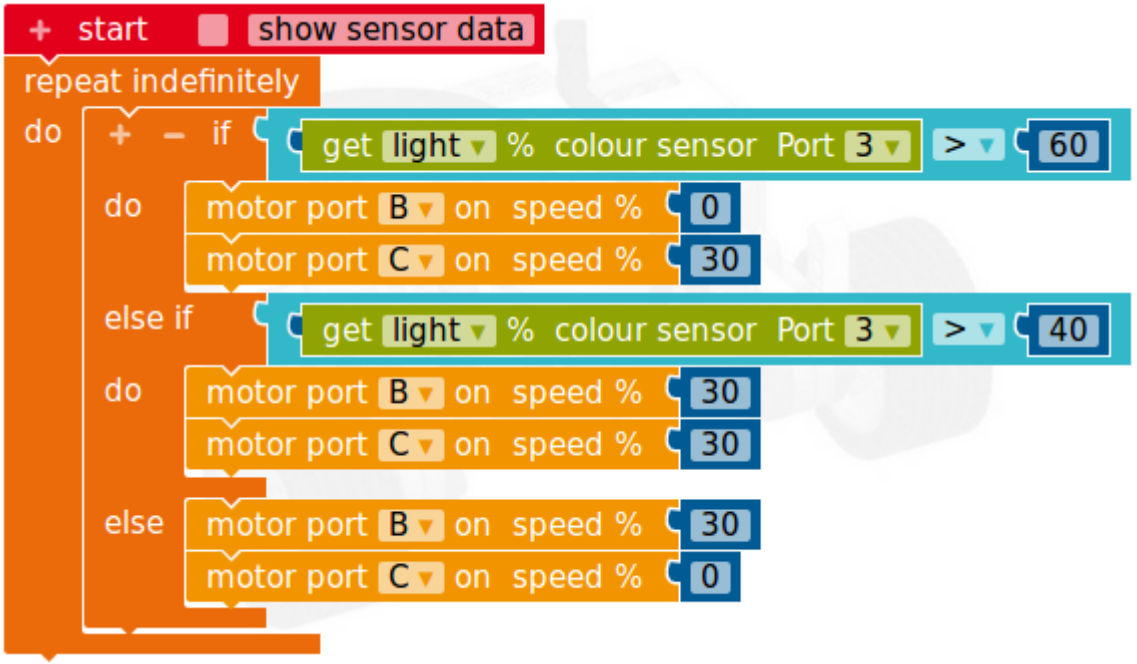
- Problem:
  - Movement is slow and jerky
- Why?:
  - Robot ONLY move left and right. It never goes straight.

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



3 States Algorithm



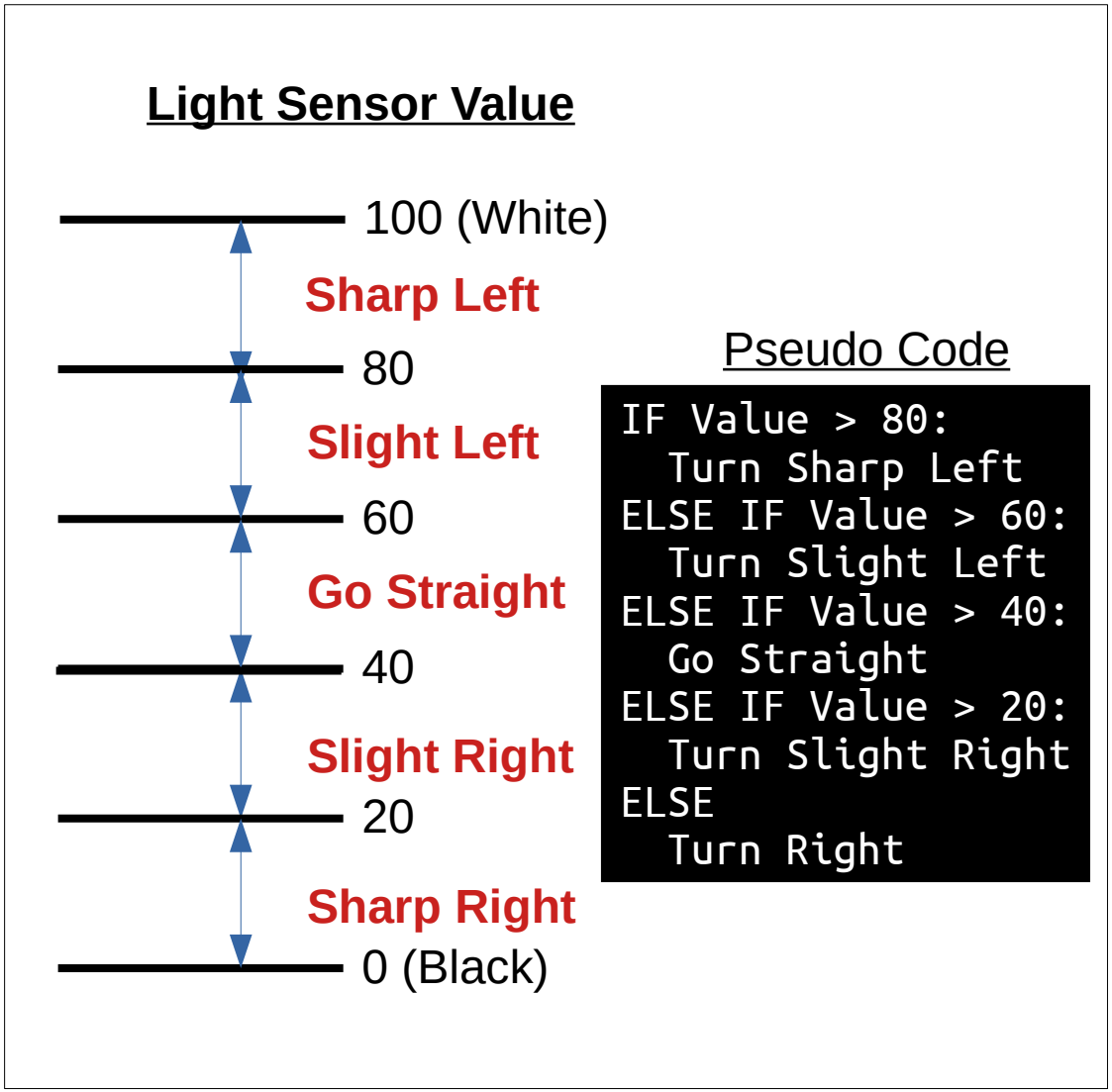

```
while True:
    if sensor.value > 60:
        lMotor.speed_sp = 0
        rMotor.speed_sp = 300
    elif sensor.value > 40:
        lMotor.speed_sp = 300
        rMotor.speed_sp = 300
    else:
        lMotor.speed_sp = 300
        rMotor.speed_sp = 0
    lMotor.run_forever()
    rMotor.run_forever()
```

Common Problems

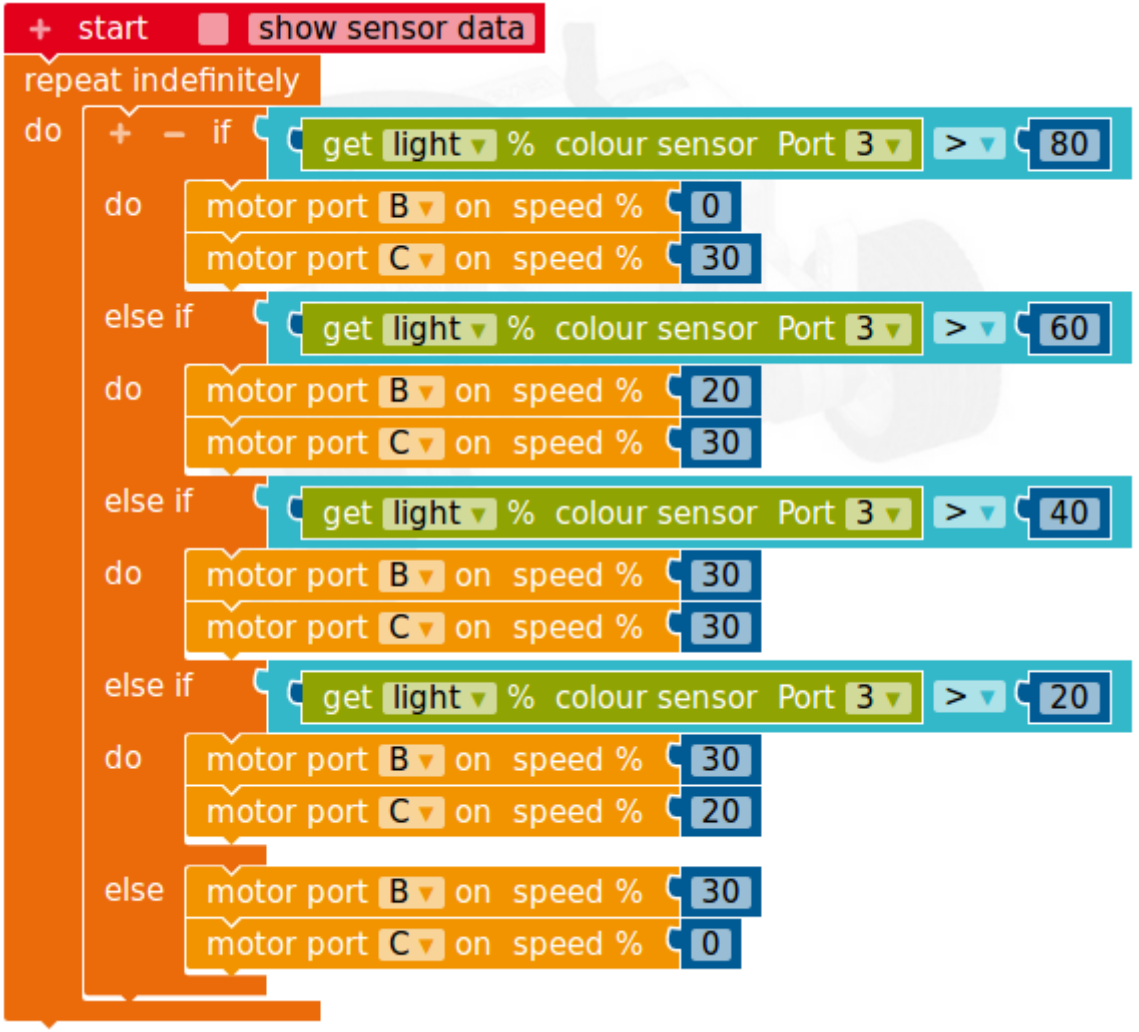
- Problem:
  - Better than 2 states, but still a little jerky
  - May be good enough
- Can we do better?

5 States Algorithm

- Take it a step further by checking for 5 levels of light sensor value:
- Robot runs even smoother than 3 states

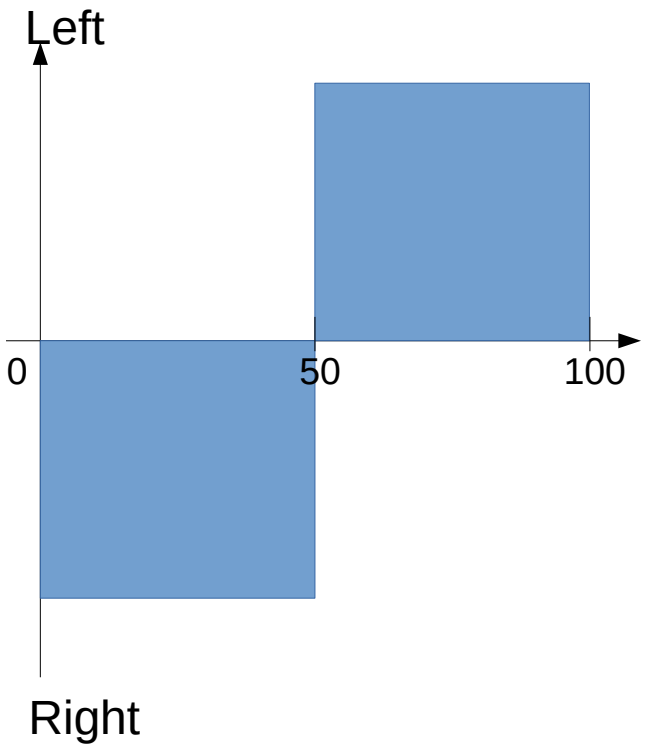


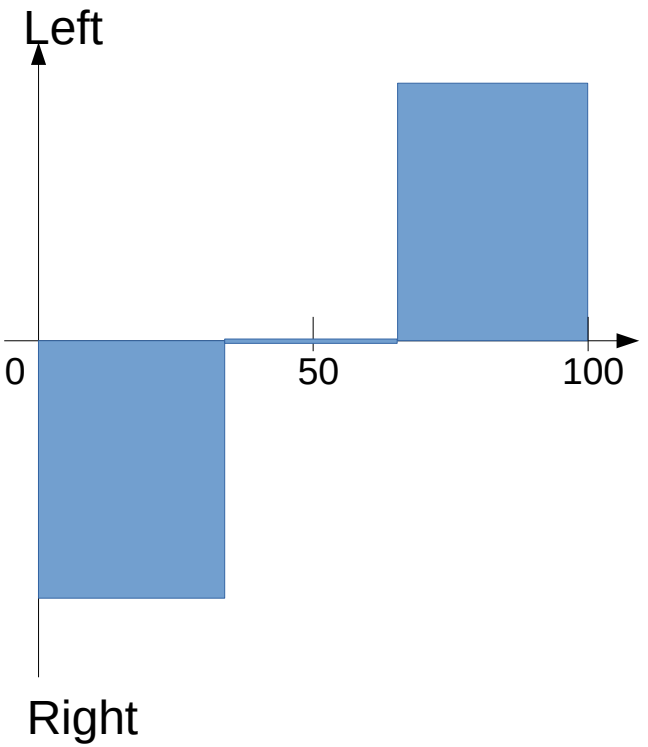
5 States Algorithm

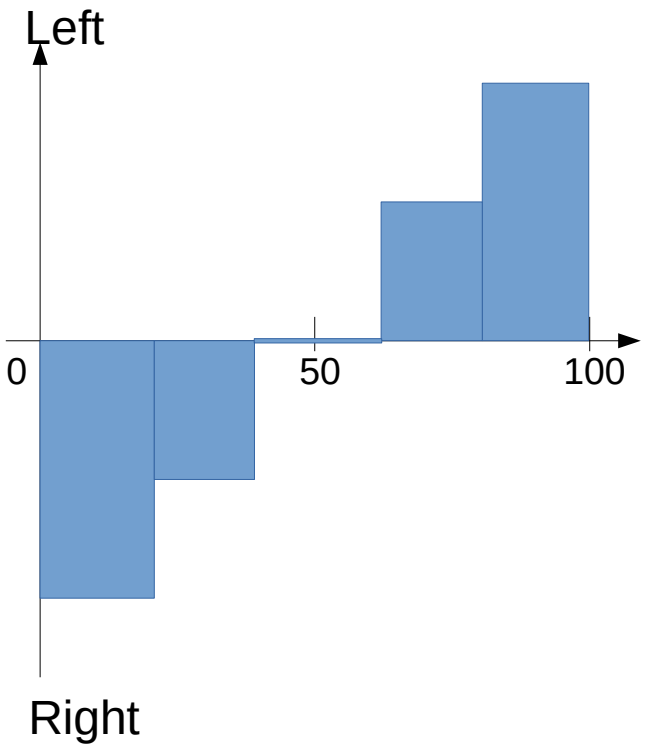



```
while True:
    if sensor.value > 80:
        lMotor.speed_sp = 0
        rMotor.speed_sp = 300
    elif sensor.value > 60:
        lMotor.speed_sp = 200
        rMotor.speed_sp = 300
    elif sensor.value > 40:
        lMotor.speed_sp = 300
        rMotor.speed_sp = 300
    elif sensor.value > 20:
        lMotor.speed_sp = 300
        rMotor.speed_sp = 200
    else:
        lMotor.speed_sp = 300
        rMotor.speed_sp = 0
    lMotor.run_forever()
    rMotor.run_forever()
```

Comparison of 2, 3, 5 states

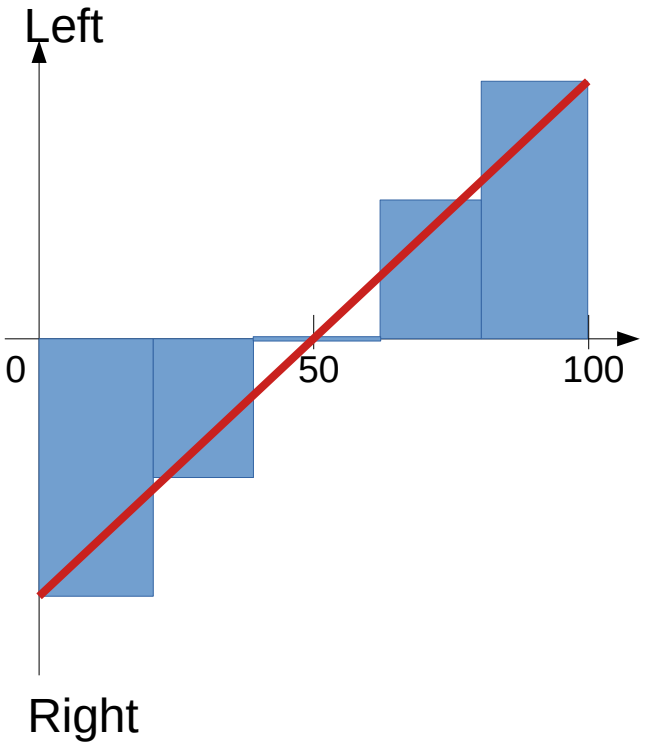






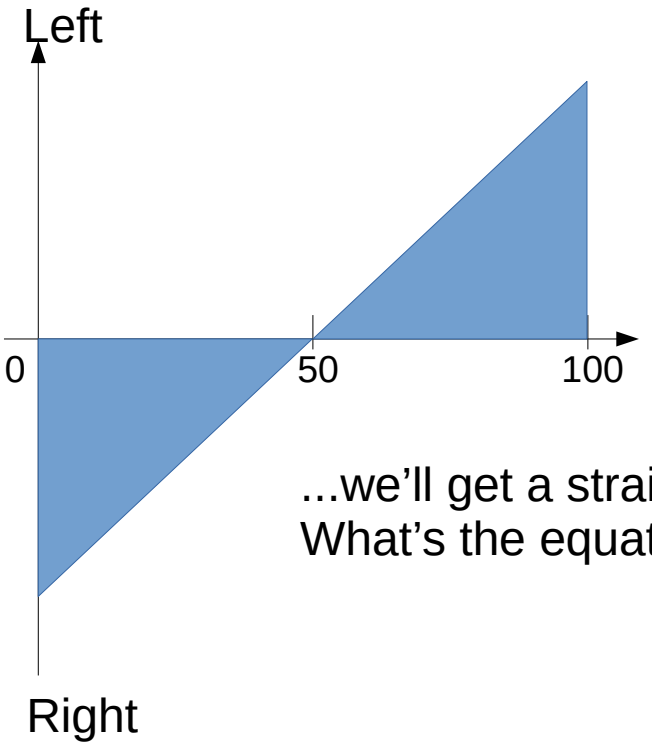
What happens if I increase the number of states?
(eg. 7 states, 9 states, 11 states)

Increasing number of states



As we increase the number of states, the diagram starts to look more like a straight line.

What if we have an infinite number of states?



...we'll get a straight line!
What's the equation of the line?

Equation of line

- Standard form

  $y = mx + c$

- Crosses x axis at x = 50, y = 0

  $0 = m(50) + c$

  $m = -c / 50$

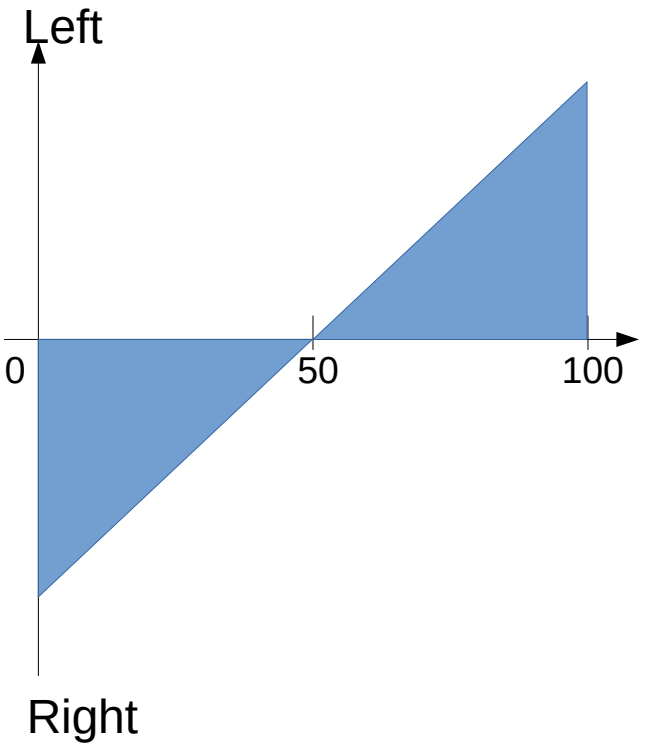
- Substitute and rearrange

  $y = (-c / 50)x + c$

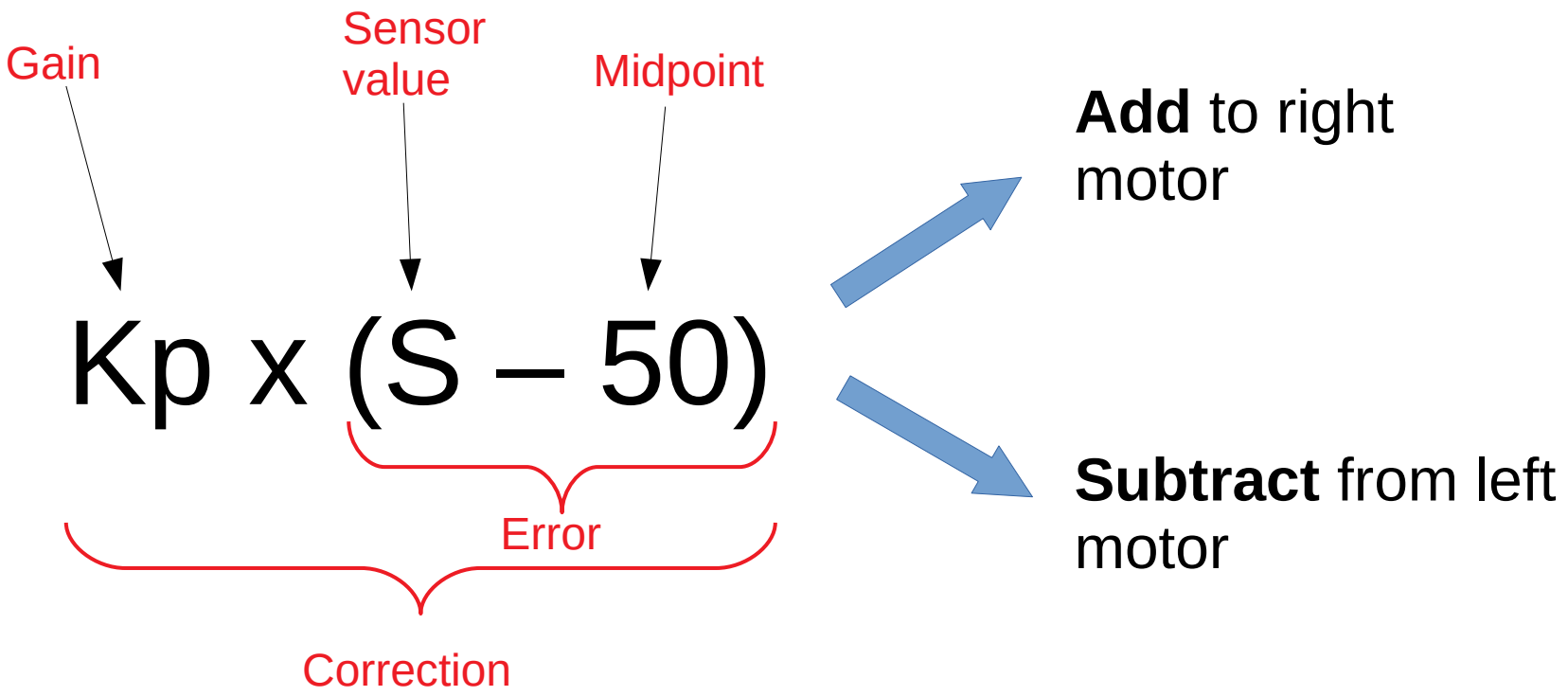
  $y = -c (x / 50 – 1)$

  $y = -c / 50 (x – 50)$

  $y = k (x – 50)$, where $k = -c / 50$



Equation of line (Engineering Style)

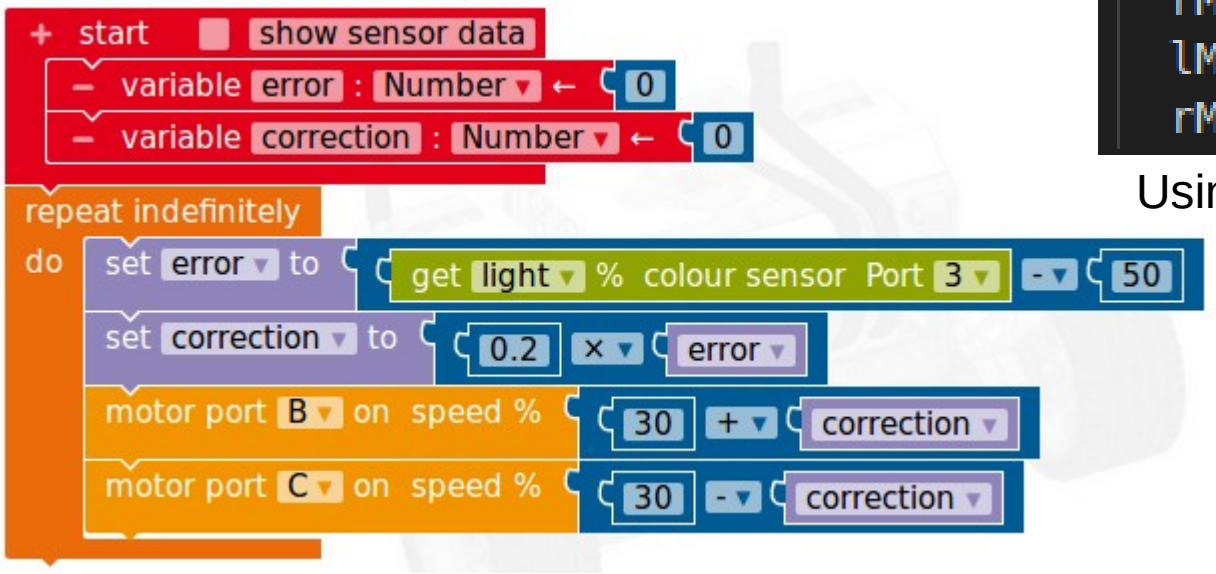


These are standard engineering terminology. Professional engineers uses these terms to make themselves sound smarter. You should do the same!

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

Proportional Control

Speeds up at sharp turns...




```
while True:
    error = sensor.value - 50
    correction = 2 * sensor.value
    lMotor.speed_sp = 300 + correction
    rMotor.speed_sp = 300 - correction
    lMotor.run_forever()
    rMotor.run_forever()
```

Using single motors

...use this if possible!

```
while True:
    error = sensor.value - 50
    correction = 1 * sensor.value
    steering_drive.on(correction, SpeedPercent(30))
```

Using move steering

Proportional Control

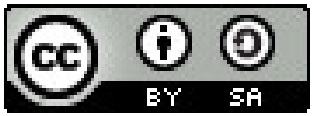
- Changing Gain:
  - Increase: Turns more sharply, may wobble
  - Decrease: Tuns more smoothly, may fail at sharp turns
- Is proportional control the best solution?
  - Depends. Proportional controls have a **straight line** response, and you **can only tune the Gain** (gradient of the line)
  - High gain may wobble too much, low gain may fail at sharp turns. **Depending on the map and robot, there may not exist a Gain value that is both smooth and can handle sharp turns.**

Proportional Control

- Test to find the best gain!
  - Suggest testing within the range of 0.1 to 4
- Possibilities to explore:
  - Non-proportional control (ie. not a straight line eqn).
    - Will a quadratic eqn work? (spoiler: No it won't, but why not?)
    - What about a cubic eqn?
  - Add in Integral and Derivative terms to make it a PID controller

Copyright

- Created by A Posteriori LLP
- Visit http://aposteriori.com.sg/ for more tips and tutorials
-



A POSTERIORI

Play · Experience · Learn