ADVANCED EV3 PROGRAMMING LESSON

EV3 Classroom:
PID Line Follower

EV3 CLASSROOM LESSON
BY EV3LESSONS.COM

Lesson Objectives

- Learn the limitations of proportional control
- Learn what PID means
- Learn how to program PID and how to tune
- Prerequisites: Math Blocks, Color Sensor Calibration, Data Wires, Variables, Proportional Control

We highly recommend knowledge of Algebra at a minimum. PID is a calculus-based concept and students should understand why it is used and the math behind it before using it.

Please use Presentation Mode as there are lots of animations.

When does Proportional Control Have Trouble?

What would a human do?

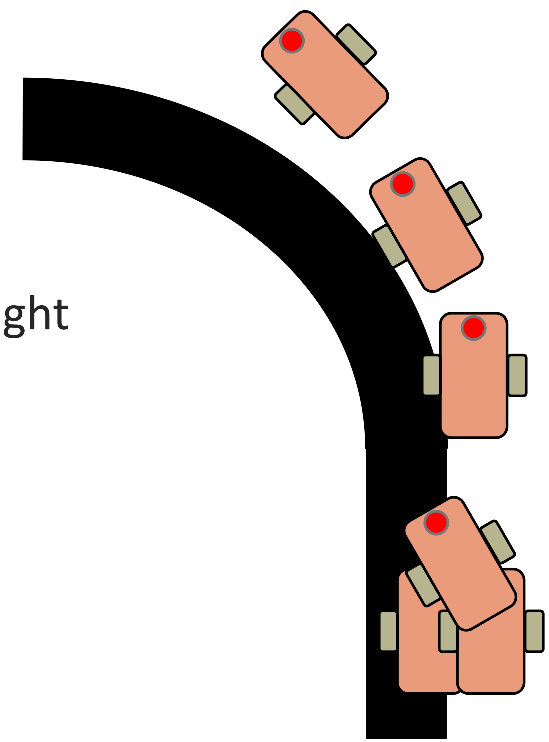
On line → go straight

On white → turn left

Moving across line → turn right

On white → turn left

Getting further from line → turn even more!



What would proportional control do?

On line → go straight

On white → turn left

Moving across line → go straight!

On white → turn left

Getting further from line → turn left the same amount!

LIGHT READING = 100%

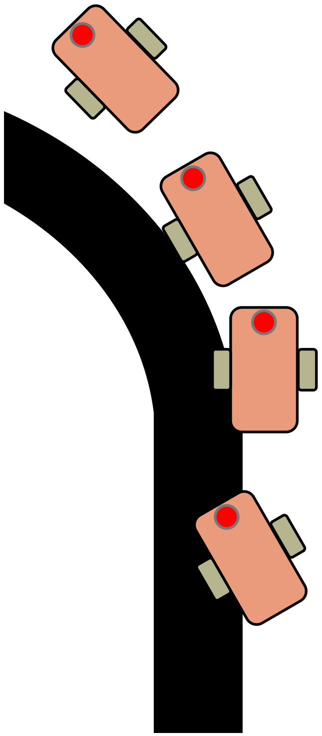
How can we fix Proportional Control?

What would a human do?

Turning left/on line → turn right

Getting further from line → turn even more!

1. Predict what the next sensor reading will be



What would proportional control do?

Turning left/on line → go straight!

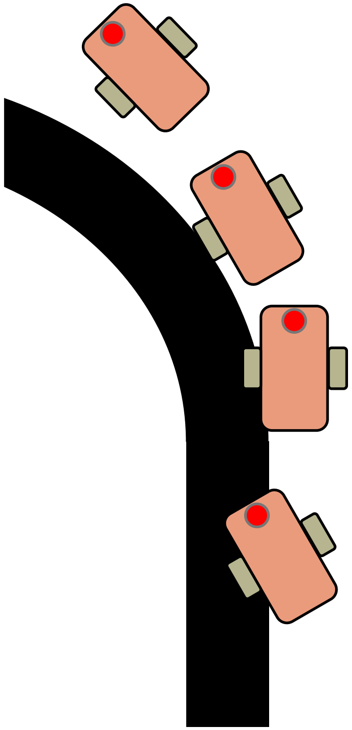
Getting further from line → turn left the same amount!

2. Has past steering fixes helped reduce error?

Integrals and Derivatives

1. Predict what the next sensor reading will be?

- If readings are: 75, 65, 55 → what do you think the next reading will be?
  - What if the readings were 57, 56, 55…
- What information did you use to guess?
- Derivative ➔ the rate at which a value is changing



2. Have past steering fixes helped reduce error?

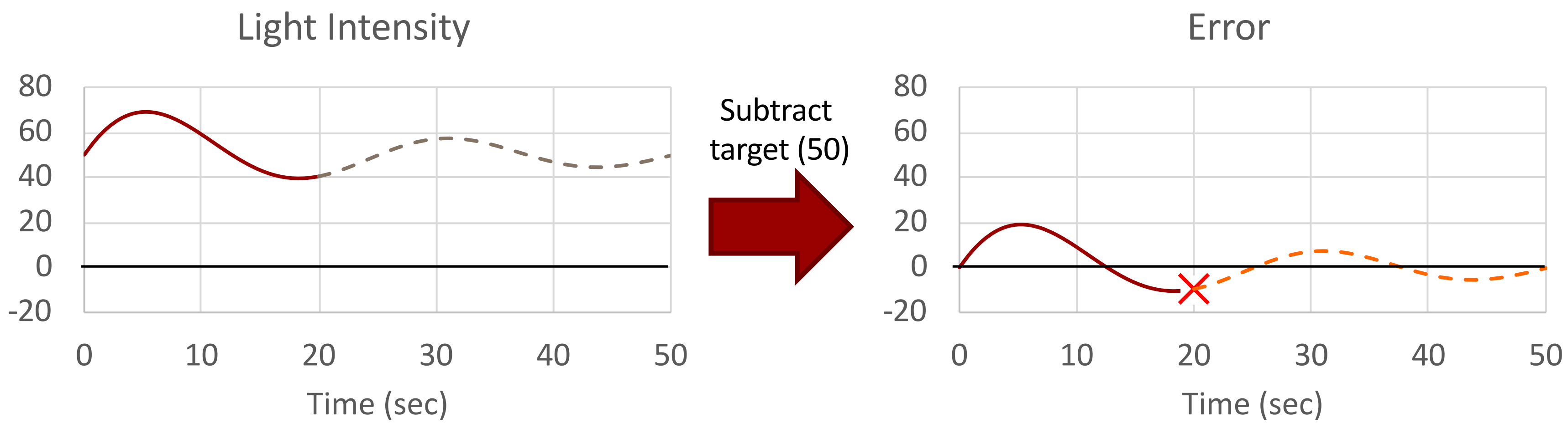
- When the correction is working well, what does error readings look like?
  - +5, -6, +4 -3…. i.e. bouncing around 0
- When steering is not working, what does error look like?
  - +5, +5, +6, +5… i.e. always on one side of 0
- How can we detect this easily?
  - Hint: look at the sum of all past errors
- What is an ideal value for this sum? What does it mean if the sum is large?
- Integral ➔ the “sum” of values

What is PID?

- **P**roportional [Error] → How bad is the situation now?
- **I**ntegral → Have my past fixes helped fix things?
- **D**erivative → How is the situation changing?
- PID control → combine the error, integral and derivative values to decide how to steer the robot

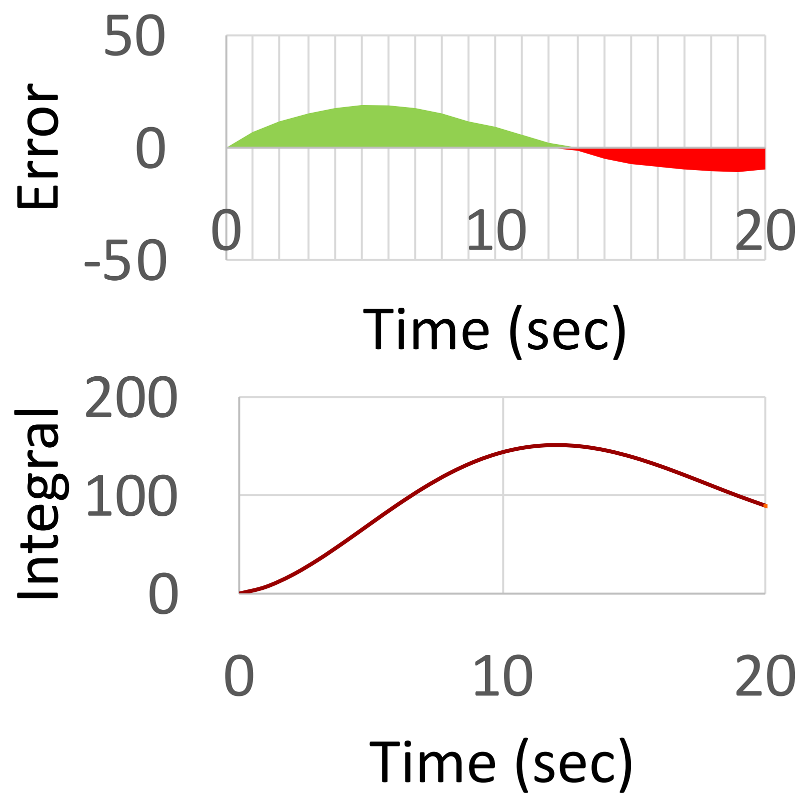
Error

- Solid line represents what you have seen, dotted line is the future
- At time 20, you see light reading = 40 and error = -10 (red X)



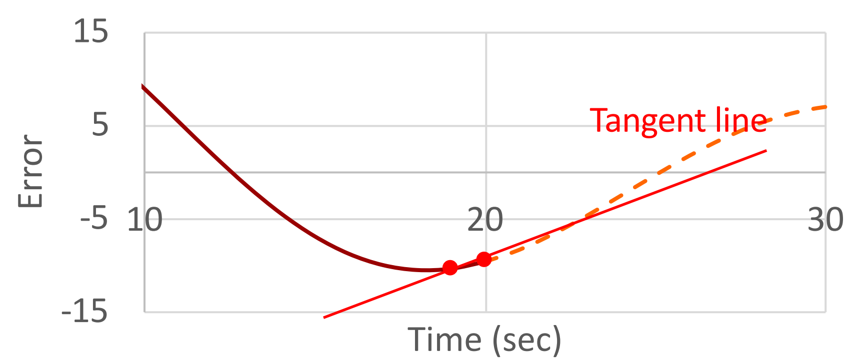
Integral

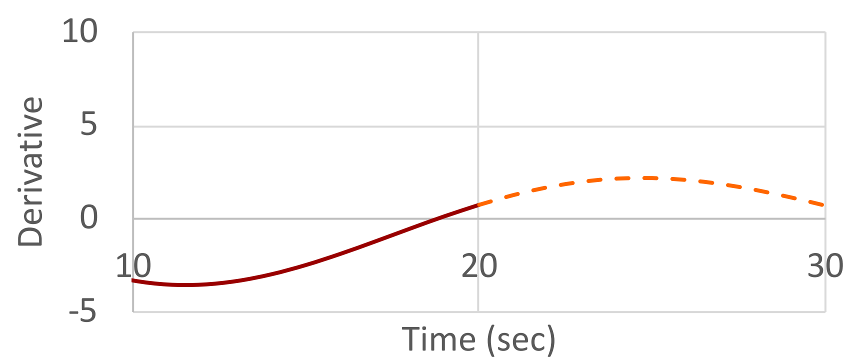
- Looks at past history of line follower
- Sum of past error
- Like area under the curve in graph (integral)
  - Green = positive area
  - Red = negative area



Derivative

- How quickly is position changing?
  - Predicts where the robot will be in the immediate future
  - Same as how fast is error changing
- Can be measured using tangent line to measurements → derivative
  - Approximated using two nearby points on graph





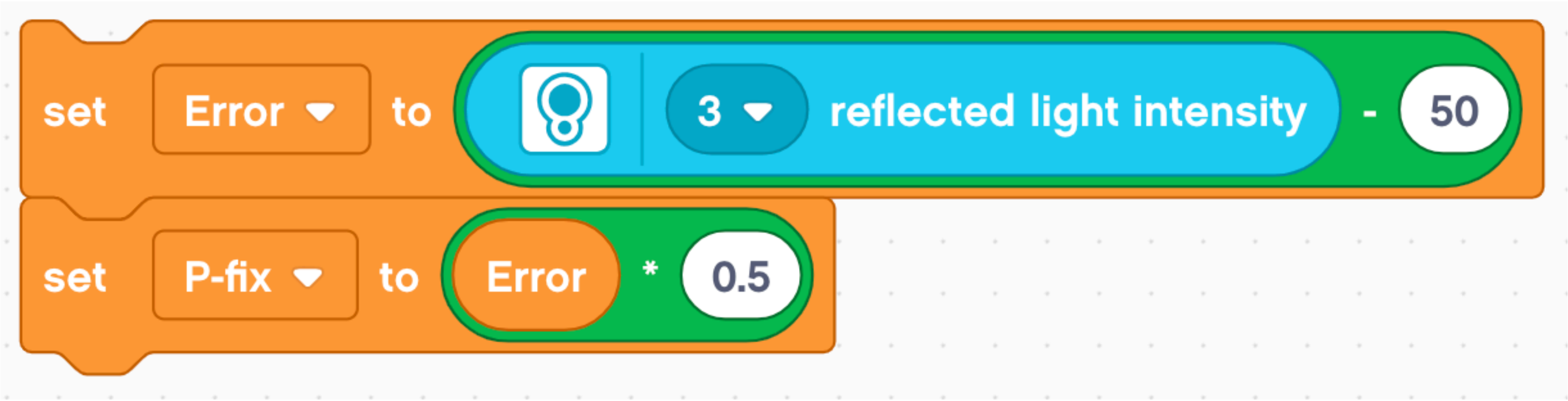
Pseudocode

1. Take a new light sensor reading
2. Compute the “error”
3. Scale error to determine contribution to steering update (proportional control)
4. Use error to update integral (sum of all past errors)
5. Scale integral to determine contribution to steering update (integral control)
6. Use error to update derivative (difference from last error)
7. Scale derivative to determine contribution to steering update (derivative control)
8. Combine P, I, and D feedback and steer robot

Code - Proportional

- This is the same as the proportional control code

Error = distance from line = reading - target

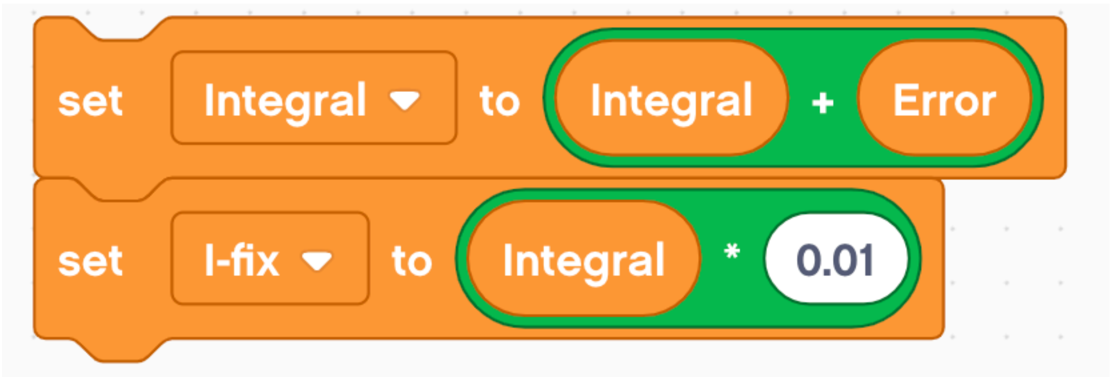


Correction (P_fix) = Error scaled by proportional constant (K_p) = 0.5

Code - Integral

- This section calculates the integral. It adds the current error to a variable that has the sum of all the previous errors.
- The scaling constant is usually small since Integral can be large

Integral = sum of all past errors = last integral + newest error

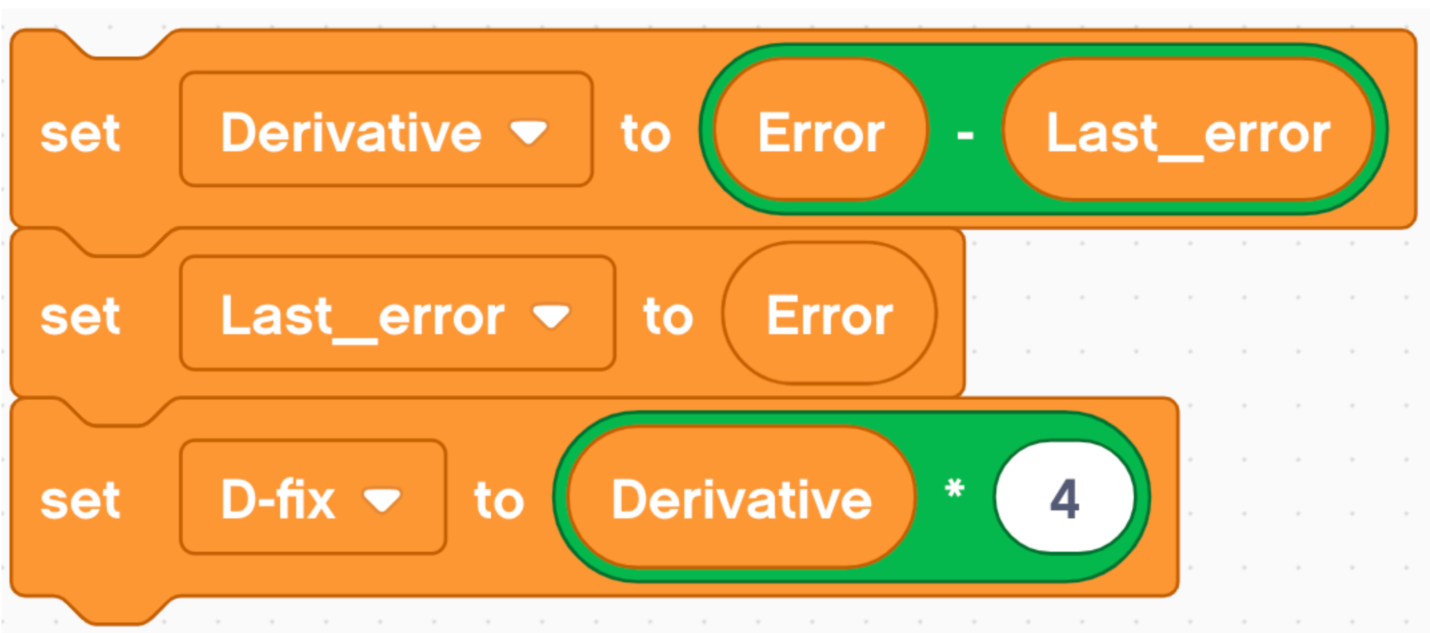


Correction (I_fix) = Integral scaled by proportional constant (K_i) = 0.01

Code - Derivative

- This section of code calculates the derivative. It subtracts the current error from the past error to find the change in error.

Derivative = rate of change of error = current error – last error



Correction (D_fix) = Derivative scaled by proportional constant (K_d) = 4.0

Putting it all Together

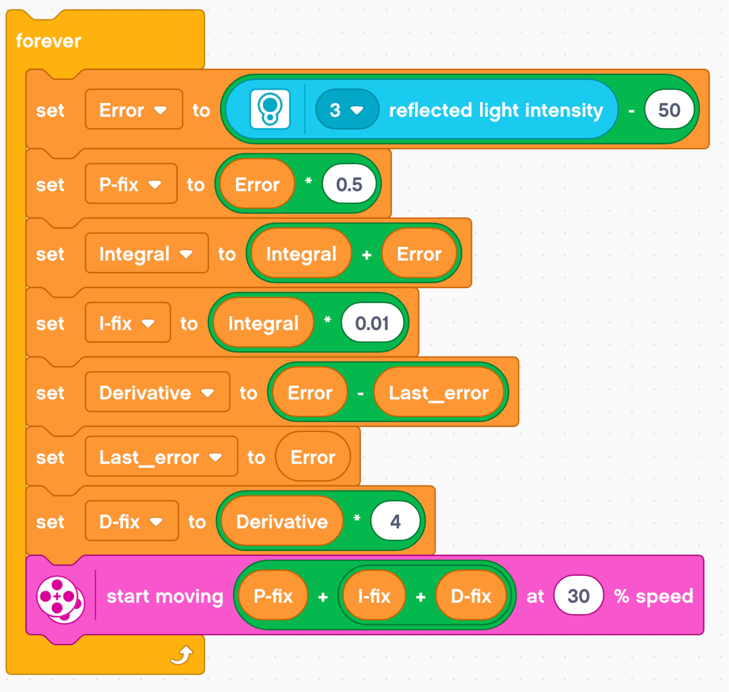
- Each of the components have already been scaled. At this point we can simply add them together.

Apply the correction the the steering of a move steering block

Add the three fixes for P, I, and D together. This will compute the final correction

Full Code

- This is what you get if you put all these parts together.
- We hope you now understand how PID works a bit better.



Proportional

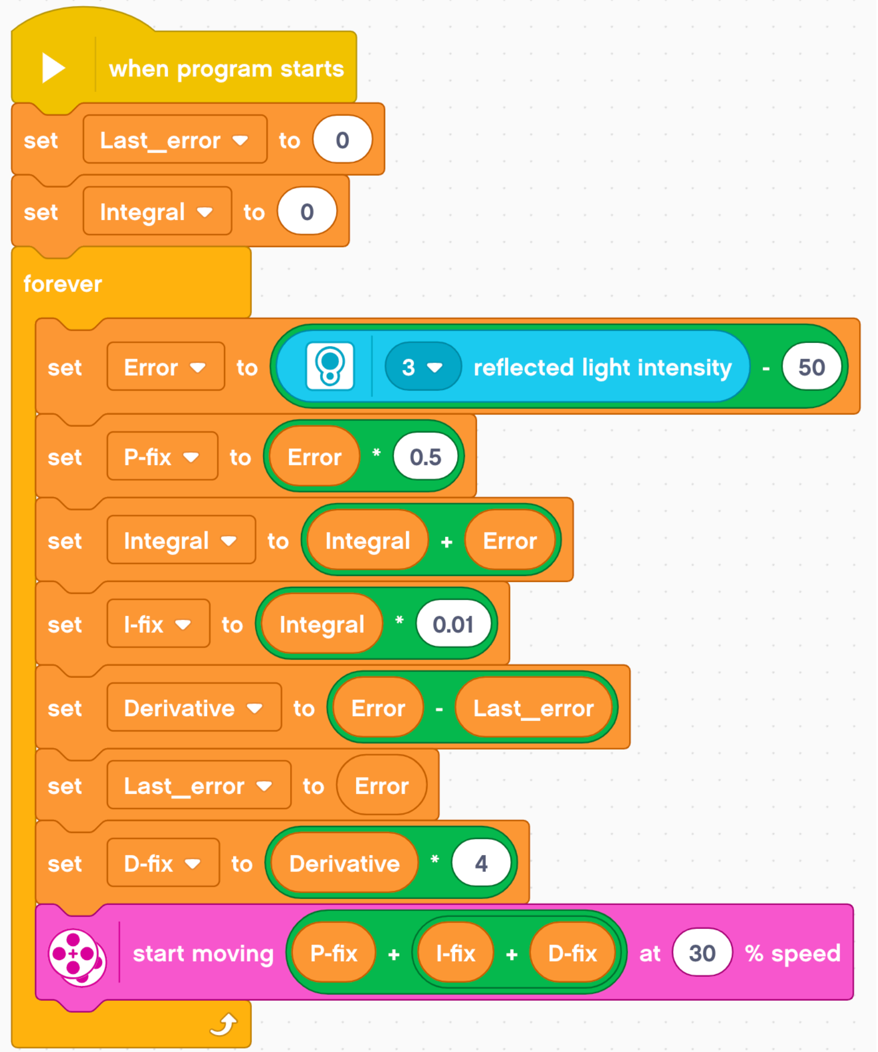
Integral

Derivative

Putting it all Together

Full Code

Set up the variables for the last error and integral before the loop and initialize to 0 because they are read before being written.



Proportional

Integral

Derivative

Putting it all Together

Key Step: Tuning The PID constants

- The most common way to tune your PID constants is trial and error.
- This can take time. Here are some tips:
  - Disable everything but the proportional part (set the other constants to zero). Adjust just the proportional constant until robot follows the line well.
  - Then, enable the integral and adjust until it provides good performance on a range of lines.
  - Finally, enable the derivative and adjust until you are satisfied with the line following.
  - When enabling each segment, here are some good numbers to start with for the constants:
    - P: 1.0 adjust by ±0.5 initially and ±0.1 for fine tuning
    - I: 0.05 adjust by ±0.01 initial and ±0.005 for fine tuning
    - D: 1.0 adjust by ±0.5 initially and ±0.1 for fine tuning

Evaluating Line followers

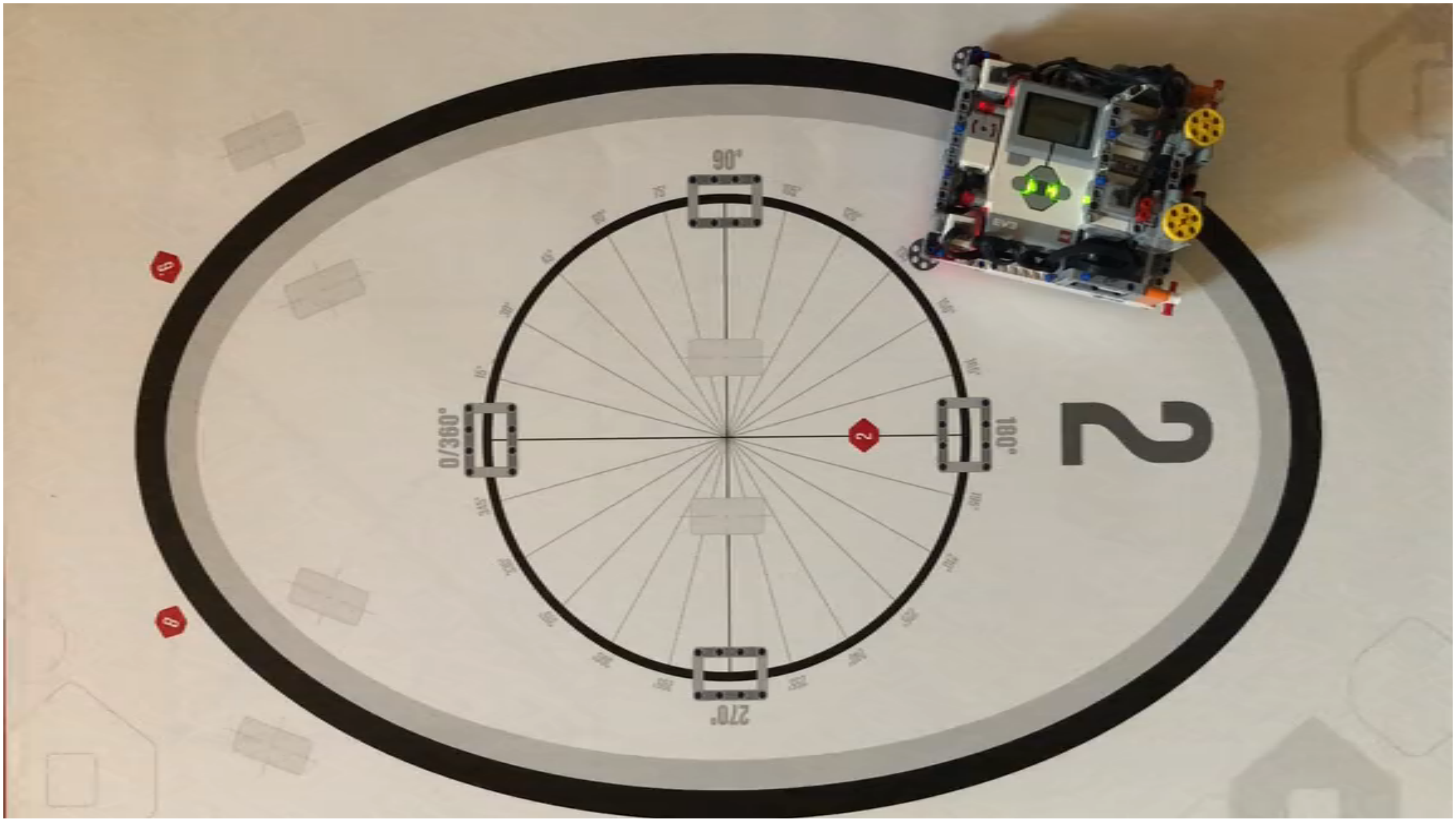
Proportional

- Uses the "P" in PID
- Makes proportional turns
- Works well on both straight and curved lines
- Good for intermediate to advanced teams → need to know math blocks and data wires

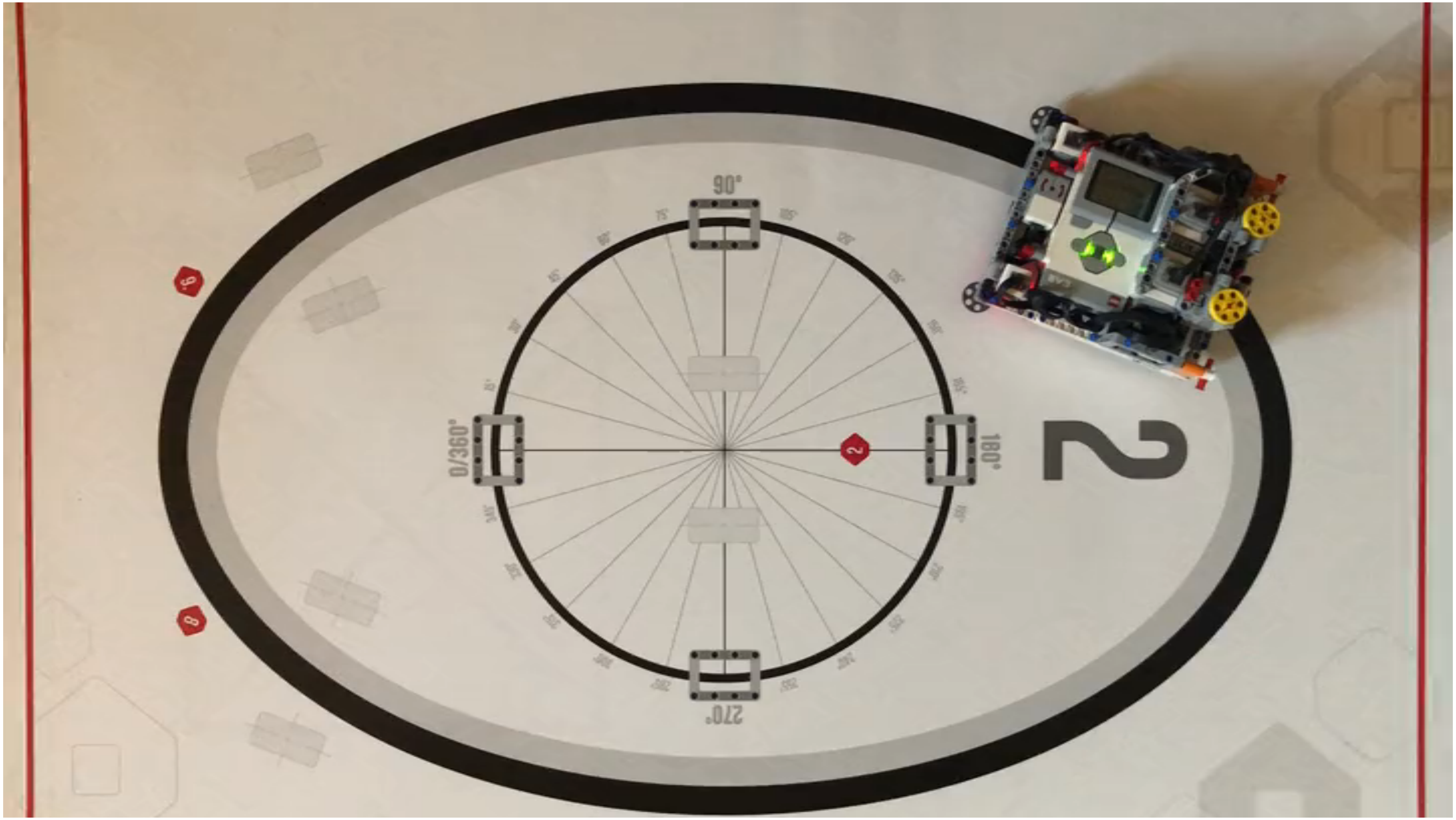
PID

- It is better than proportional control on a very curved line, as the robot adapts to the curviness
- However, for FIRST LEGO League, which mostly has straight lines, proportional control can be sufficient

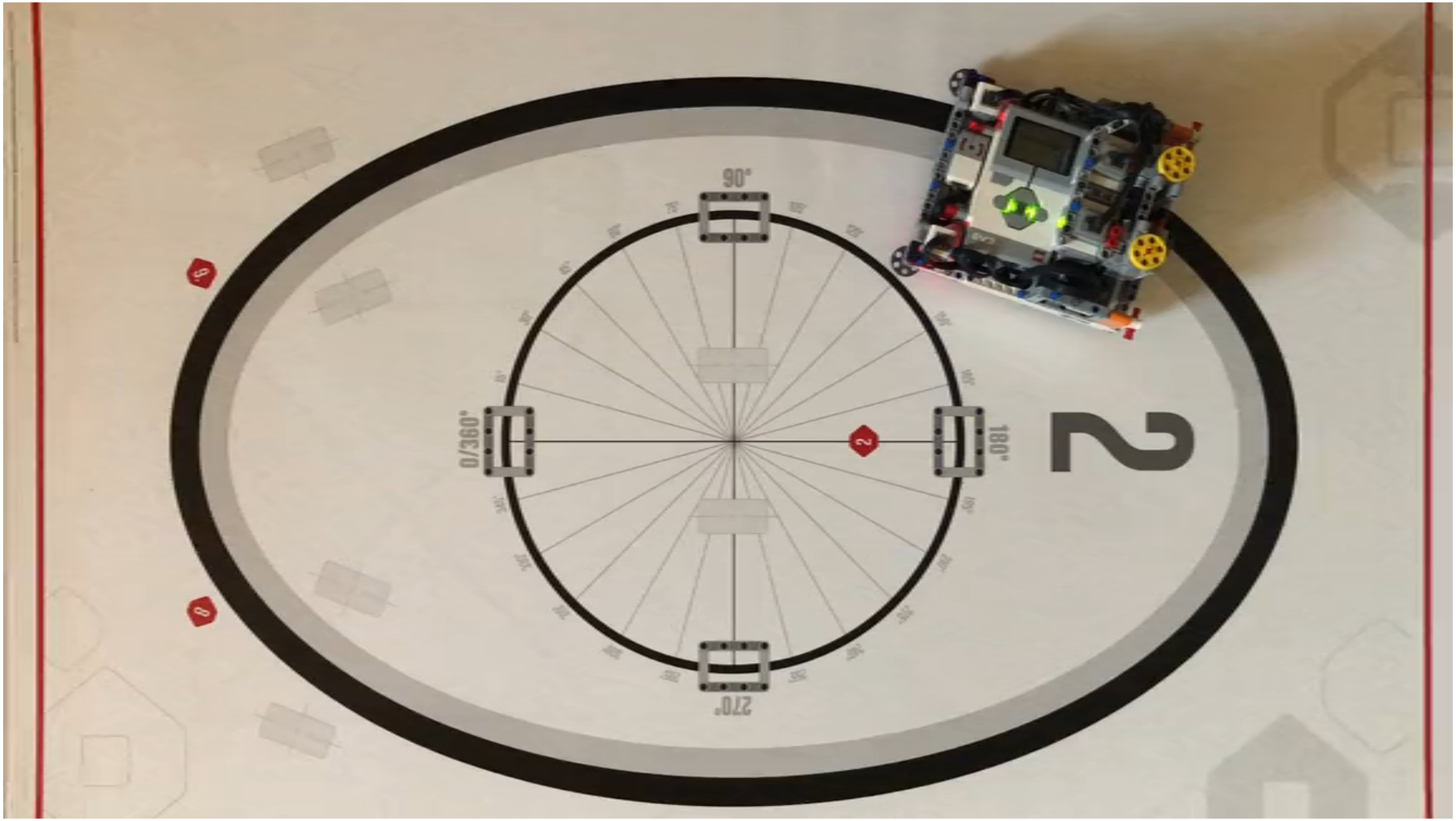
Proportional Control (0.6 Constant)



Proportional Control (0.8 Constant)



PID Control



CREDITS

- This tutorial was created by Sanjay Seshan and Arvind Seshan
- More lessons at www.ev3lessons.com