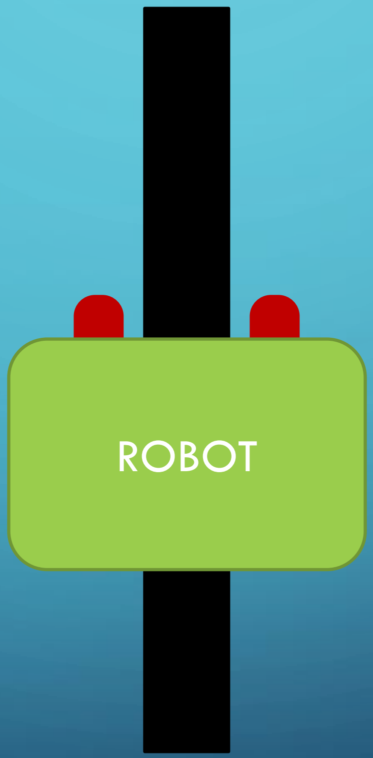
LINE FOLLOWING WITH PID

SUTD IEEE

OBJECTIVE: FOLLOW A LINE



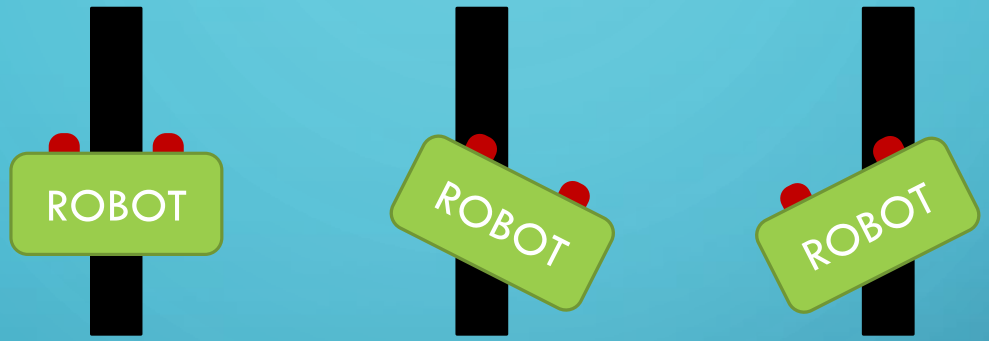
But how do you follow a line?

- Keep the line in the center!

OBJECTIVE: FOLLOW A LINE

- Open Loop System
  - Use logic to control. If this is the input → what is the output?
- Closed Loop System (With feedback)
  - Input → Decision → Output → Input → …

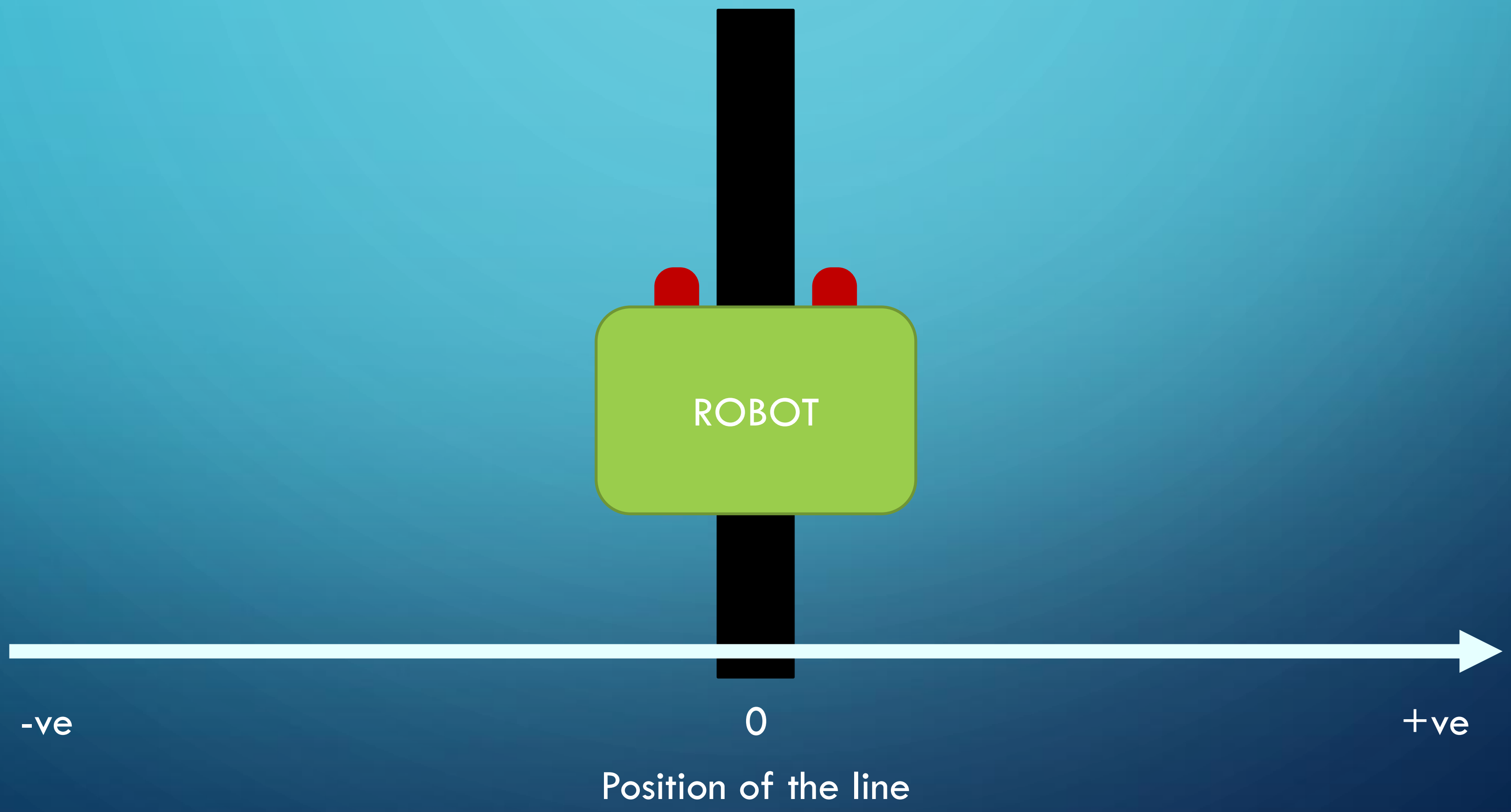
OPEN LOOP SYSTEM



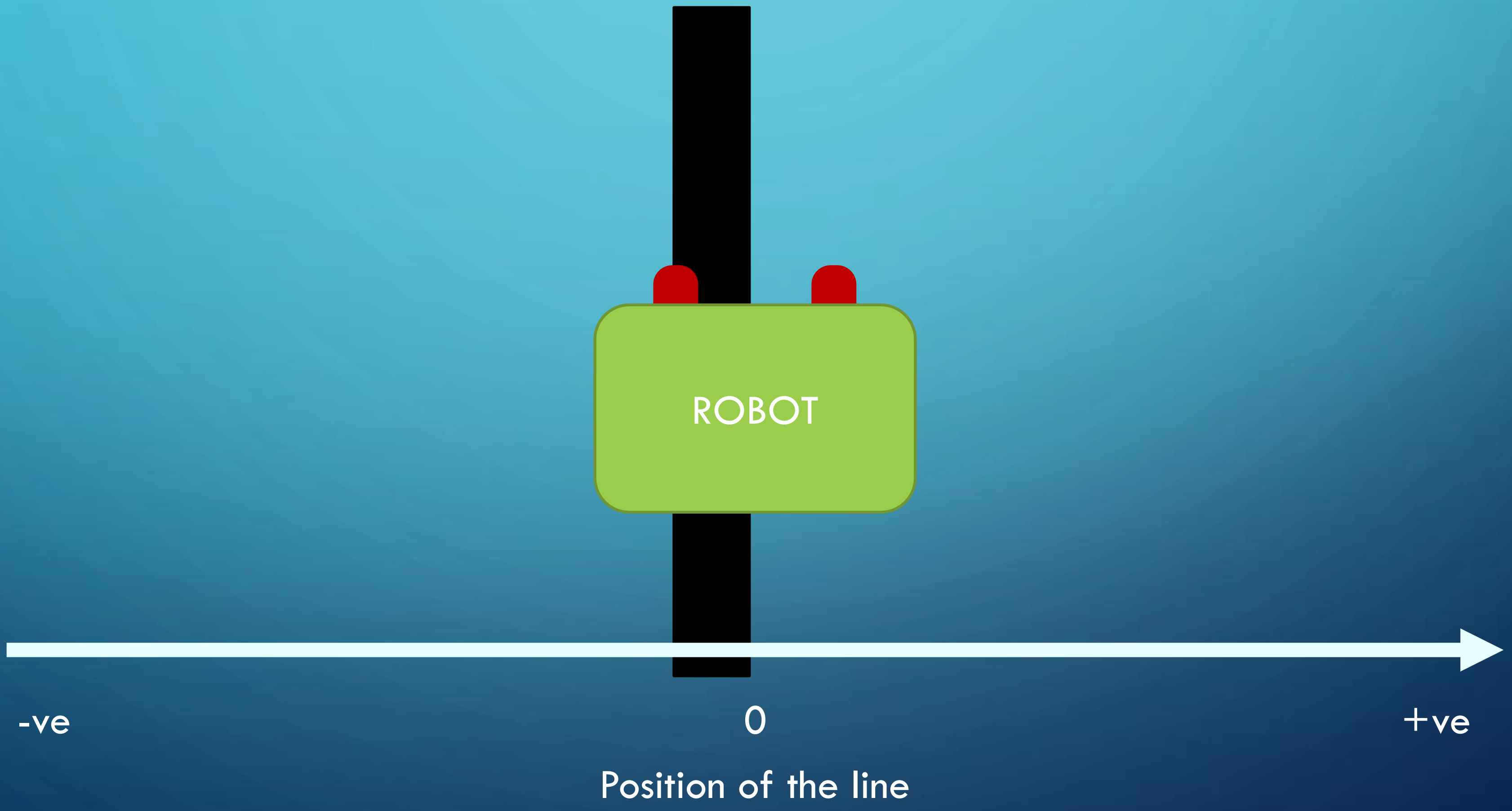
0 >> White
1 >> Black

INPUT (LEFT SENSOR | RIGHT SENSOR)	OUTPUT
0 | 0	Move Forward
1 | 0	Turn CCW (Counter Clockwise)
0 | 1	Turn CW (Clockwise)

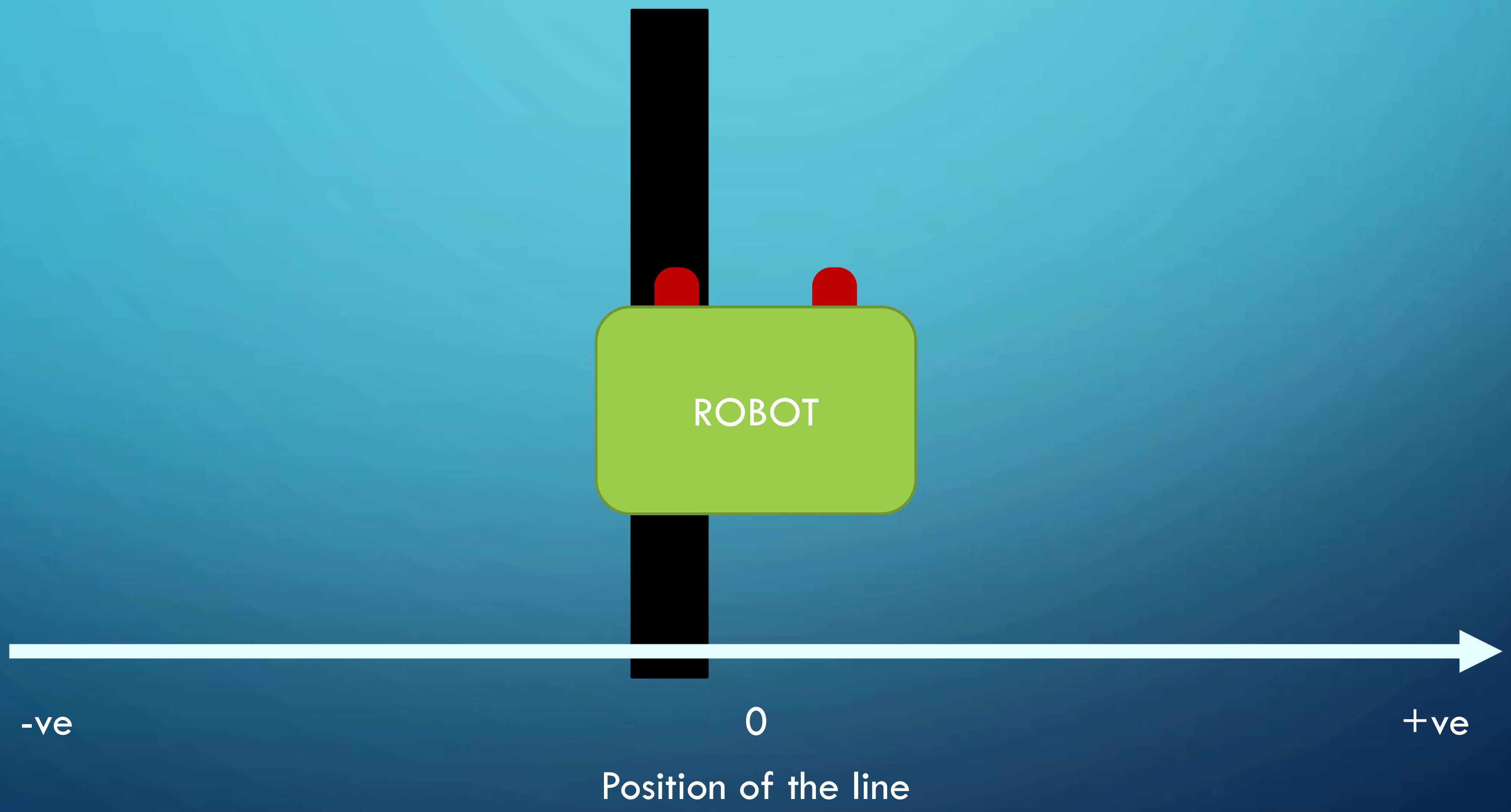
CLOSED LOOP SYSTEM



CLOSED LOOP SYSTEM



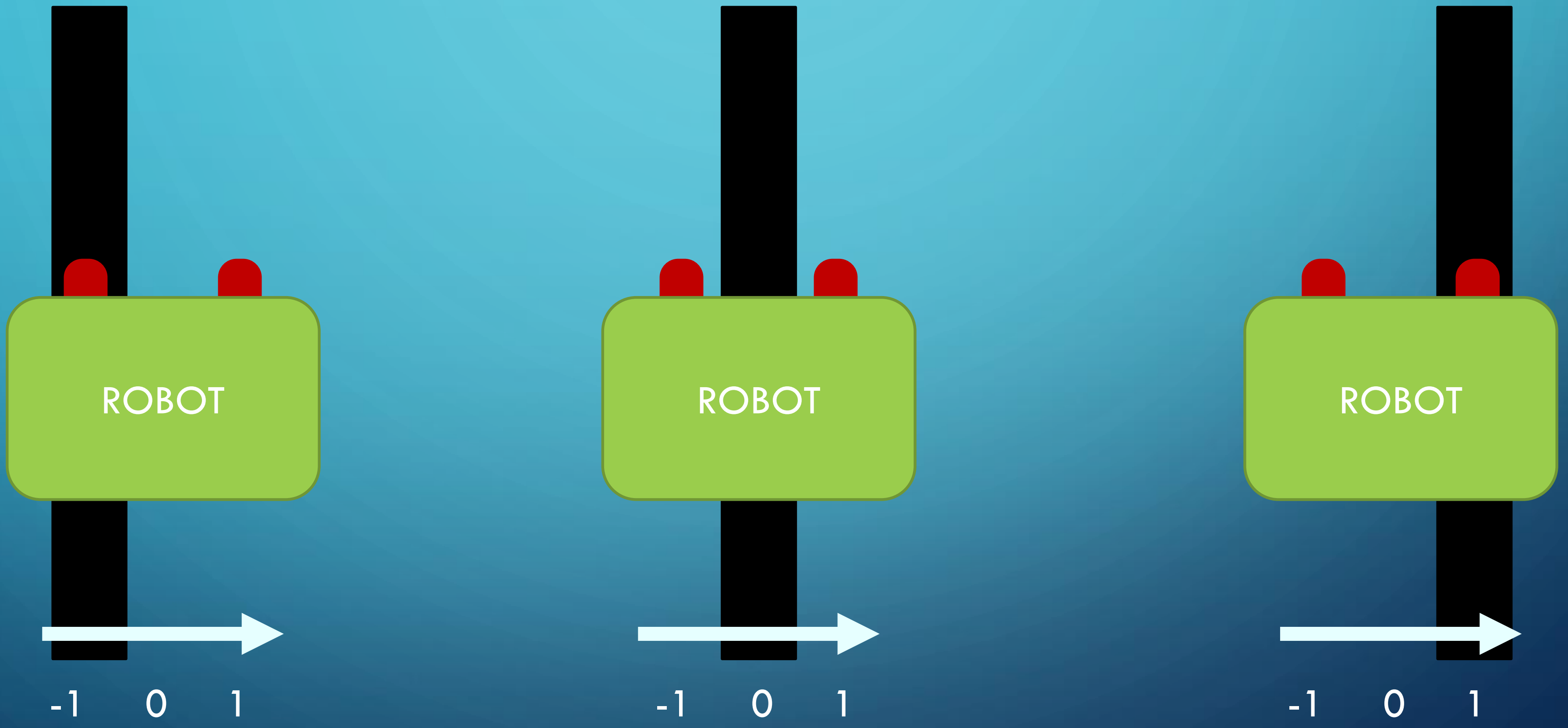
CLOSED LOOP SYSTEM



CLOSED LOOP SYSTEM

- PID
  - Proportional
  - Integral
  - Derivative
- The greater the error, the greater the response.

IMPLEMENTING PID



Position of the line

IMPLEMENTING PID: HACKING THE IR SENSOR

IMPLEMENTING PID

- Obtaining the position of the line:
  - Subtract value of left-sensor from the right-sensor

INPUT (LEFT SENSOR | RIGHT SENSOR)	ERROR (Position of Line)
0 | 0	0 – 0 = 0
1 | 0	0 – 1 = -1
0 | 1	1 – 0 = 1

IMPLEMENTING PID: P CONTROLLER

- Default state (Error = 0): Both wheels move forward at same speed
  - Robot moves forwards
- Error = 1: Left wheel spins faster than right wheel by K_P * Error
  - Robot turns right (while moving forward)
- Error = -1: Right wheel spins faster than right wheel by K_P * Error
  - Robot turns left (while moving forward)
- K_P is the Proportional constant. Usually a +ve value.
- Allows robot to react according to how large the error actually is.

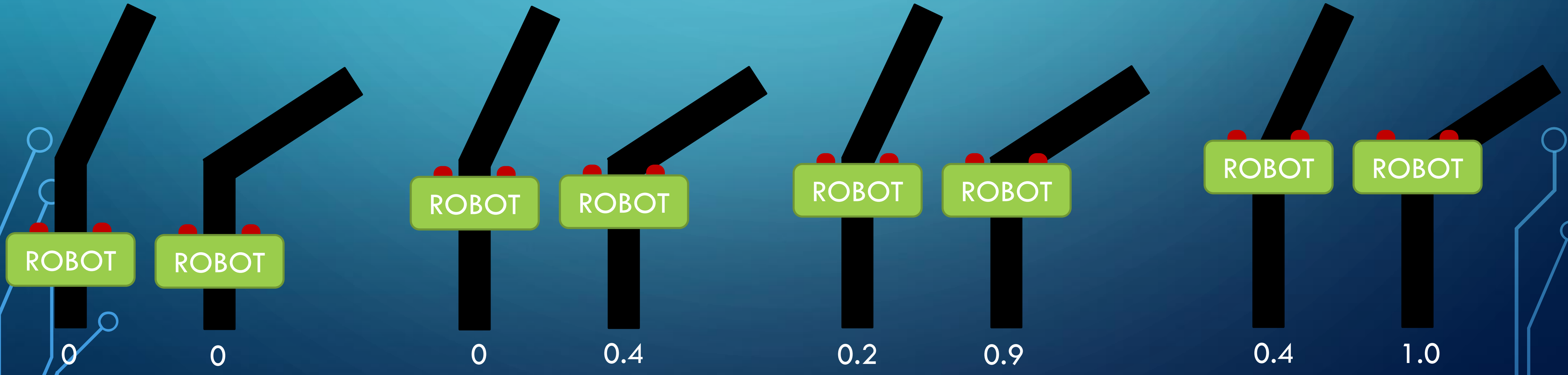
P CONTROLLER: PSEUDO CODE

```
float P {0}, I {0}, D {0};

setup():
    - Set required pins (IR sensor pins, Motor pins)

loop():
    - Read IR sensor pins >> Determine error
    - P = error
    - moveLeftMotor(BIAS + K_P * P)
    - moveRightMotor(BIAS – K_P * P)
```

BIAS –
Default motor speed so robot continuously moves forwards

IMPLEMENTING PID: I CONTROLLER

- Sum of errors over an interval
- Defines how quick the system (robot) responds to a change from the 0 point.
  - Larger change from 0 point >> Larger response
- Response proportional by K_I to Integral component



IMPLEMENTING PID: D CONTROLLER

- Prevent overshooting by predicting future error based on rate of change.
  - Higher rate of change of error >> Larger response.
- Response proportional by K_D to Derivative component

I,D CONTROLLER: PSEUDO CODE

float P {0}, I {0}, D {0};

setup():

- Set required pins (IR sensor pins, Motor pins)

loop():

- Read IR sensor pins >> Determine error
- P = error
- I = I + error
- D = error – prevError

- prevError = error
- moveLeftMotor(BIAS + (K_P * P + K_I * D + K_D * D))
- moveRightMotor(BIAS – (K_P * P + K_I * D + K_D * D))

BIAS –
Default motor speed so robot continuously moves forwards