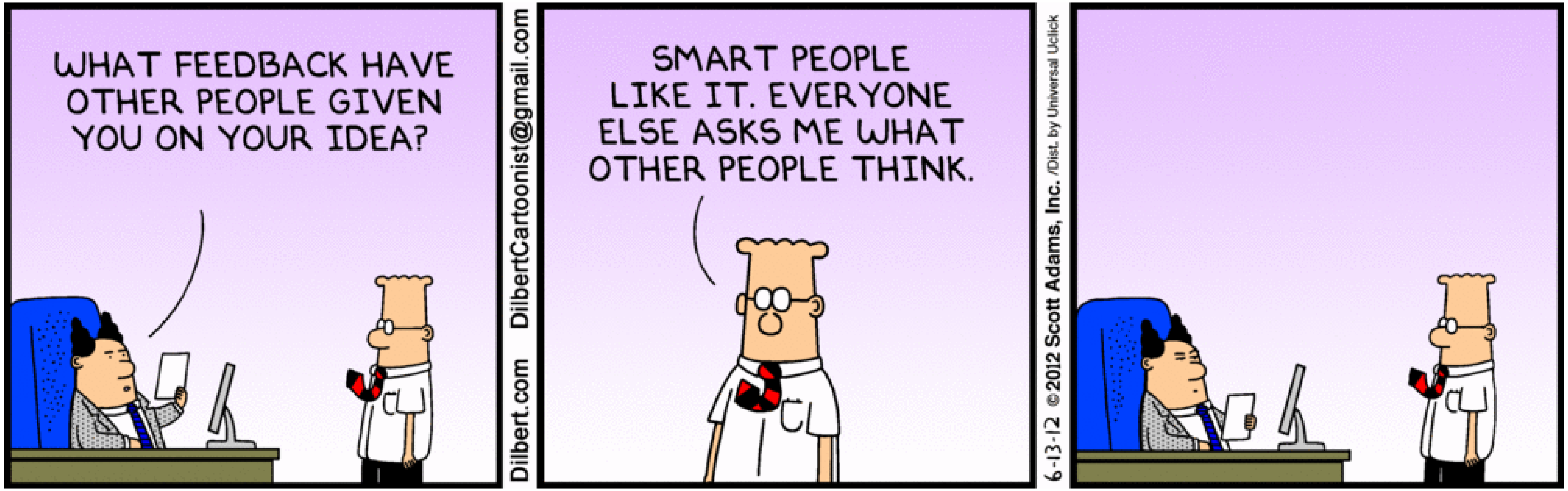
Feedback Control

Sources: 3400 lecture from Justin Selig and Adarsh Jayakumar, 2017

Based on ECE 2100/2200 knowledge

MAE 4780/5780: Feedback Control Systems

ECE 4530: Analog Integrated Circuit Design → **ECE 5540:** Advanced Analog Integrated VLSI Circuit Design

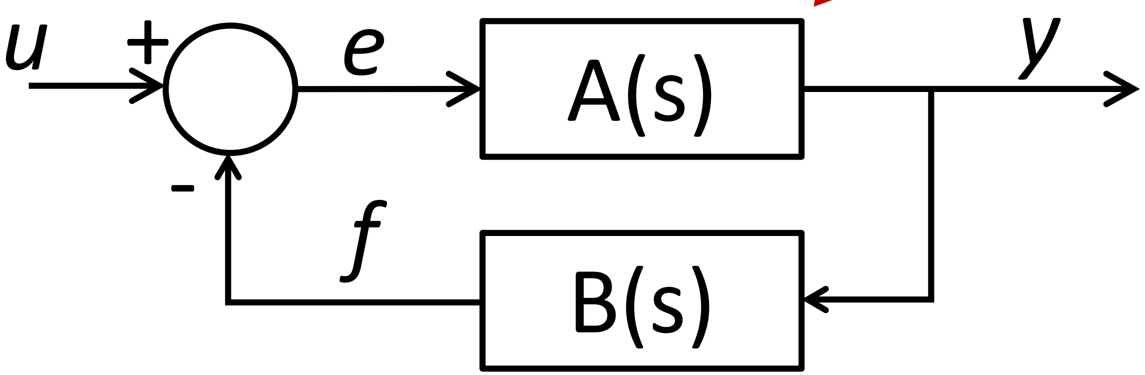


- Introduction
- EE-version
- Robot-version
- Servos!
- …and a little EE again

ECE 3400: Intelligent Physical Systems

Feedback Control

Optimizing a system's performance by feeding its output back into its input.



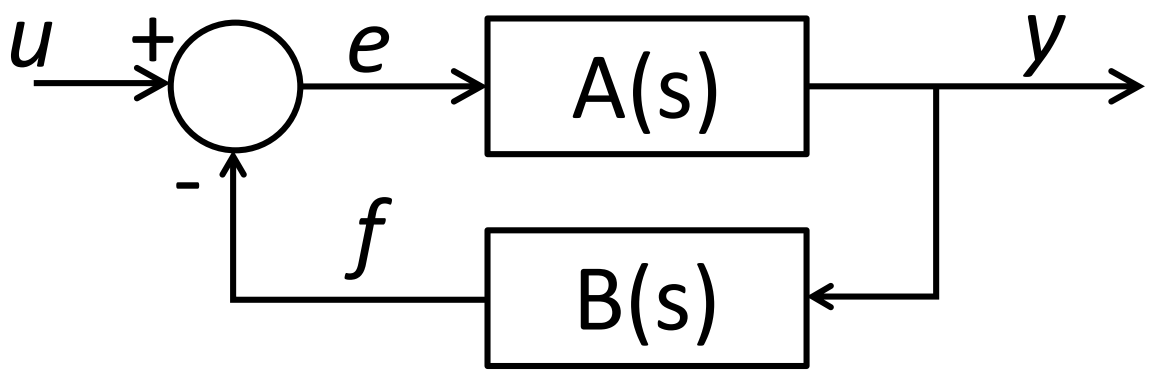
...amplifier, motor, robot, etc

Why should you care now?

- Reason about circuit design
- Used for speed control in the servos
- Better line following
- etc…

Feedback Control

Optimizing a system's performance by feeding its output back into its input.

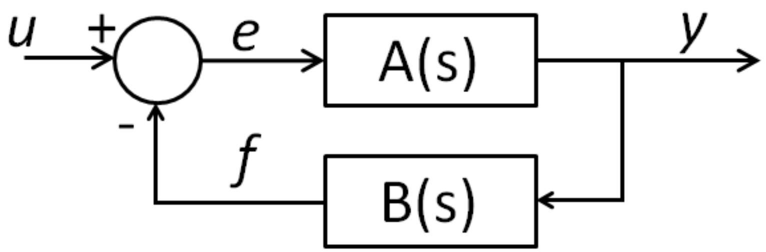


- **y = eA(s)**
- **f = yB(s)**
- **e = u - f**

$$y = eA(s) = A(s)\,[u - f] = A(s)\,[u - yB(s)] = uA(s) - yA(s)B(s)$$

$$\boxed{H(s) = \frac{y}{u} = \frac{A(s)}{1+A(s)B(s)}}$$

Feedback in a Non-Inverting OpAmp



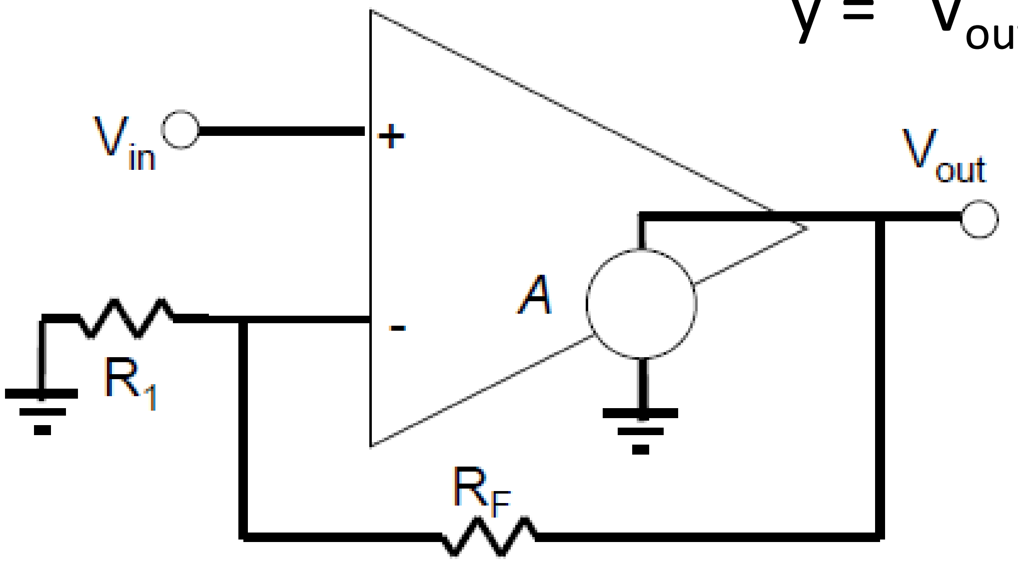
$$\boxed{H(s) = \frac{y}{u} = \frac{A(s)}{1+A(s)B(s)}}$$

Circuit Analysis

$u = V_{in}$

$y = V_{out}$

$$A(s) = A_v \qquad B(s) = \frac{R_1}{R_1 + R_f}$$

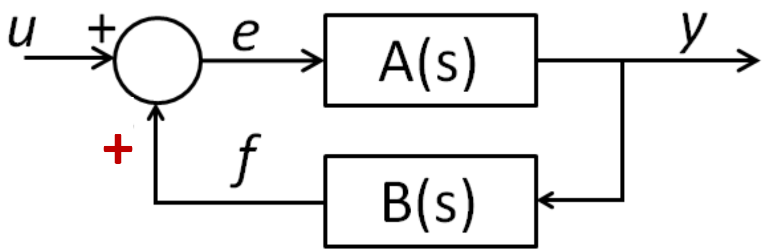


$$H(s) = \frac{V_{out}}{V_{in}} = \frac{A(s)}{1+A(s)B(s)} = \frac{A_v(R_1 + R_f)}{(R_1 + R_f + A_v R_1)}$$

For $A_v \rightarrow \infty$

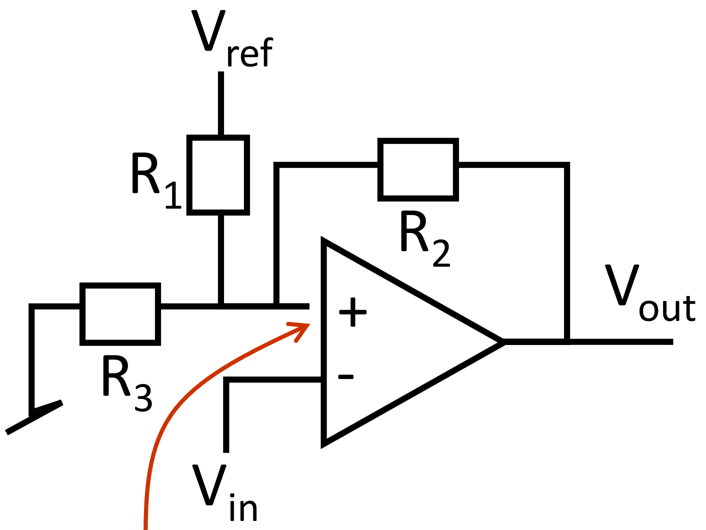
$$\frac{V_{out}}{V_{in}} = \frac{R_1}{R_1 + R_f} = 1 + \frac{R_f}{R_1}$$

Feedback in a Schmitt Trigger



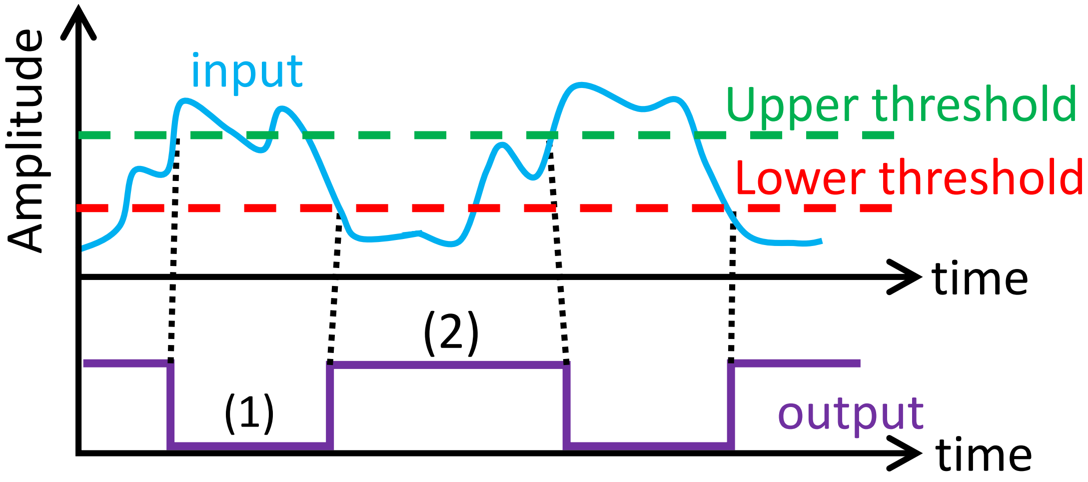
$$H(s) = \frac{y}{u} = \frac{A(s)}{1+A(s)B(s)}$$

Circuit Analysis

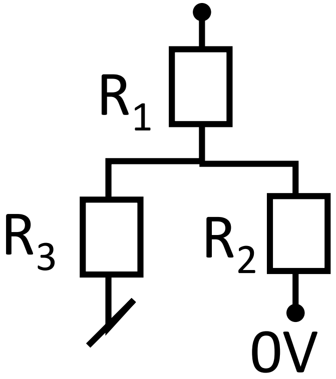


Positive feedback!

$R_1 = R_2 = R_3 = 10K$



(1)

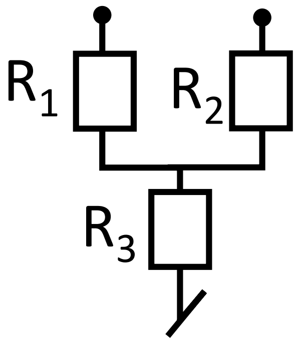


$V_{out} = 0V$

$$V_a = \frac{(R_2||R_3)V_{ref}}{R_1 + R_2||R_3}$$

$V_a = 1.66V$

(2)

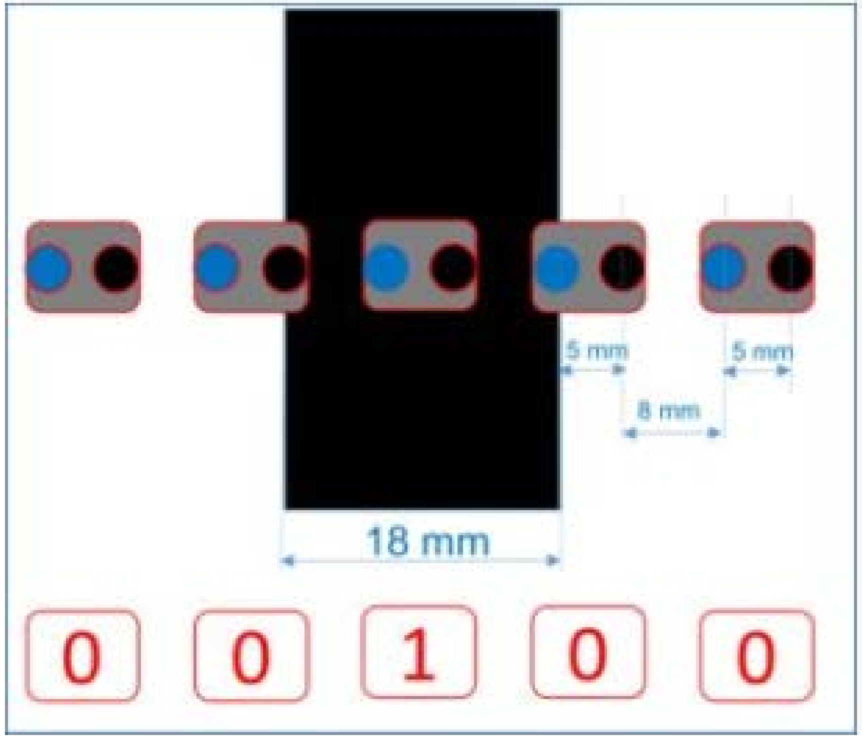


$V_{out} = 5V$

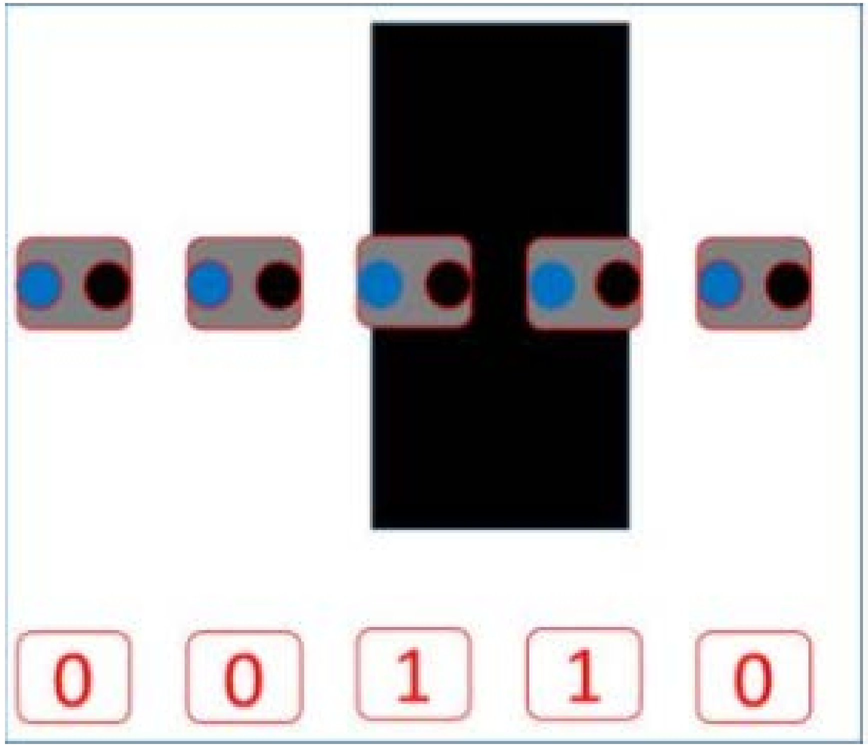
$$V_a = \frac{R_2 V_{ref}}{R_2 + R_1||R_3}$$

$V_a = 3.3V$

Feedback Control and Line Following



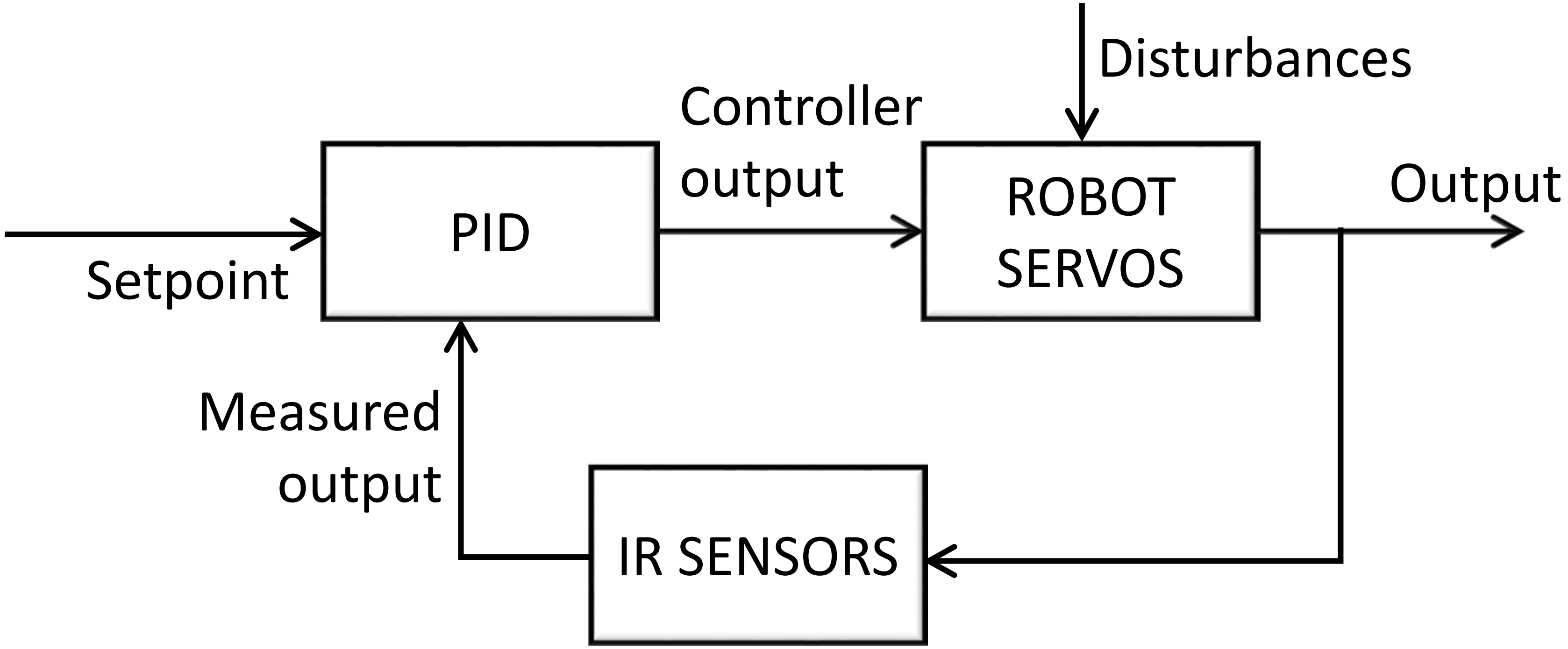
Left Servo Speed : 50
Right Servo Speed : 50



Left Servo Speed : 50 + error
Right Servo Speed : 50 - error

Feedback Control and Line Following

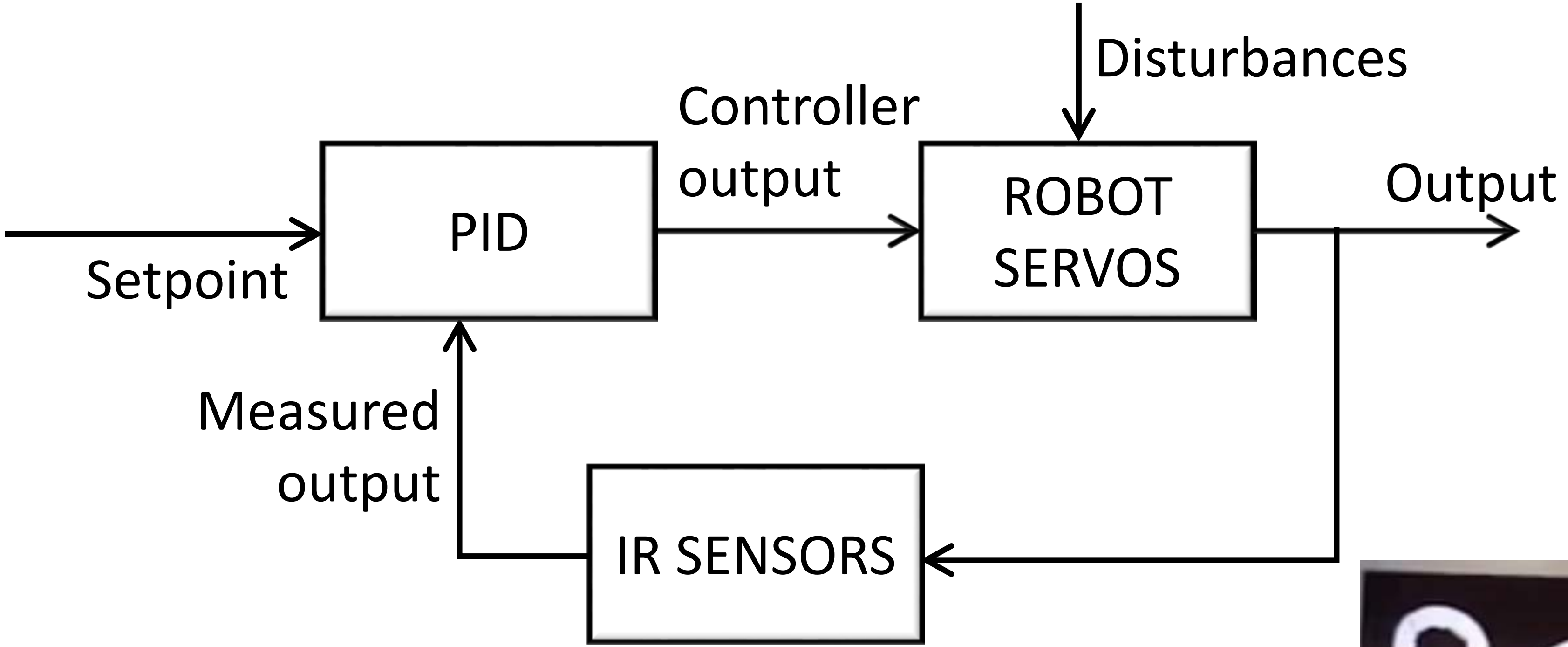
PID control



- Set-point: Distance from line we want
- Controller Output: Motor speeds we want
- Measured Output: Deviation from the line

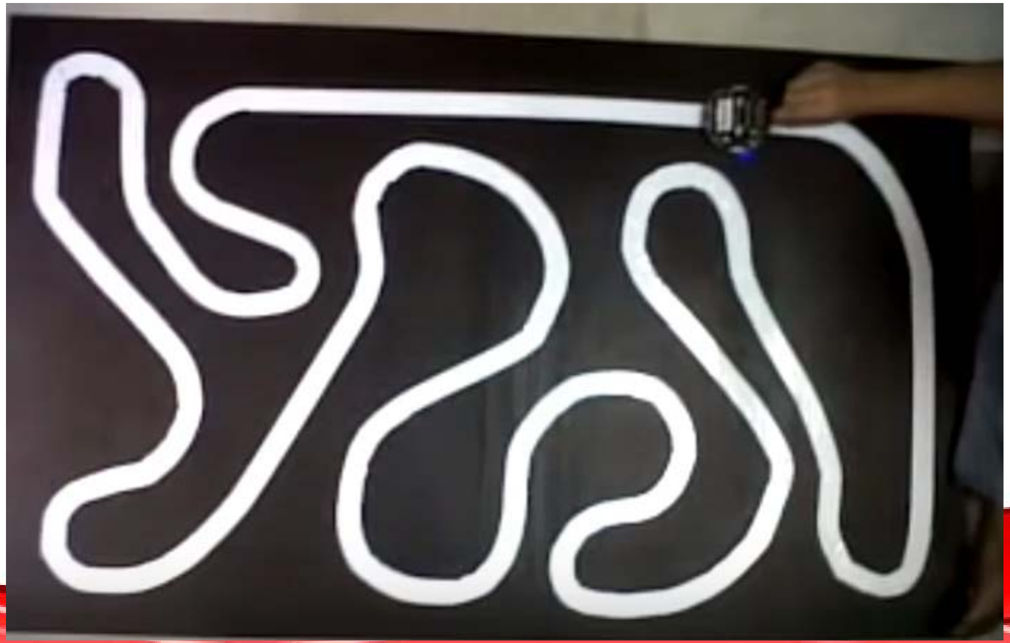
Feedback Control and Line Following

PID control



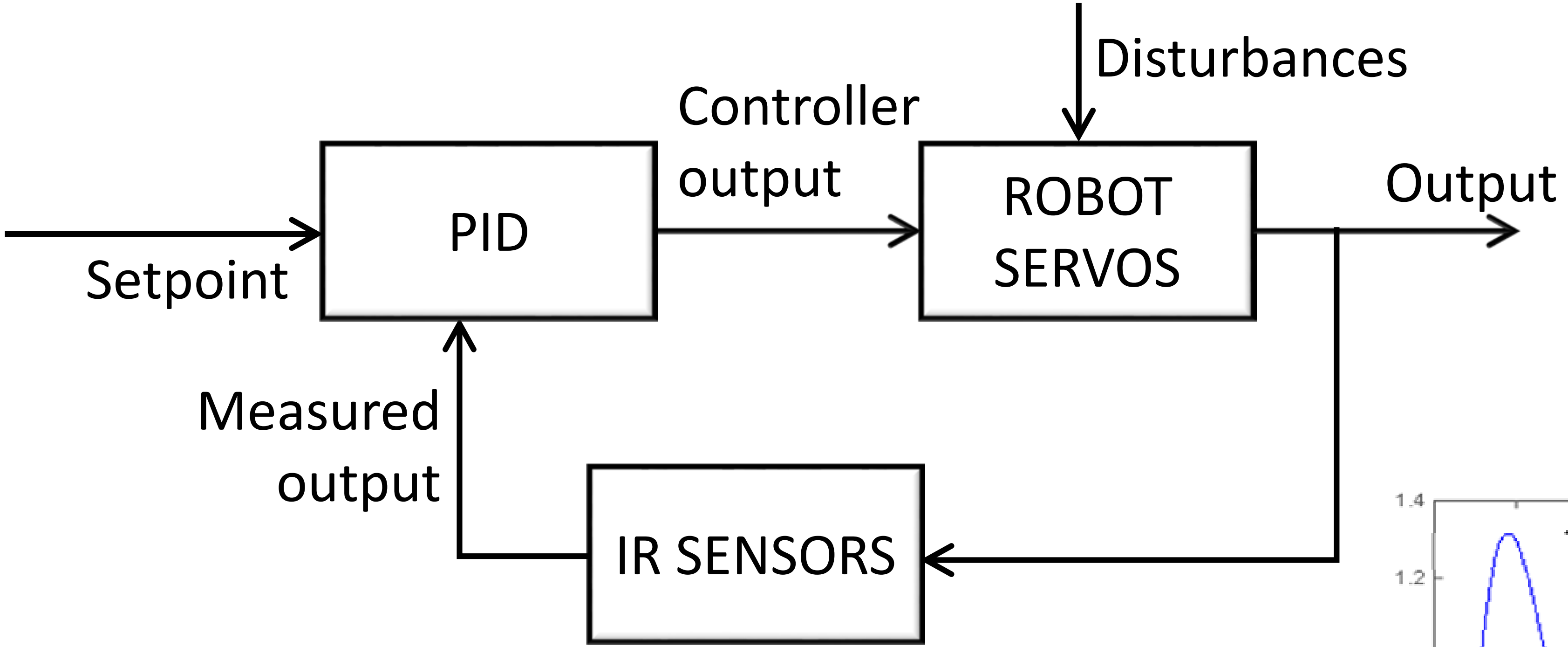
- *Proportional:* Looks at the instantaneous error
- *Integral:* Sum of errors over time
- *Derivative:* Rate of change of error over time

www.Polulu.com

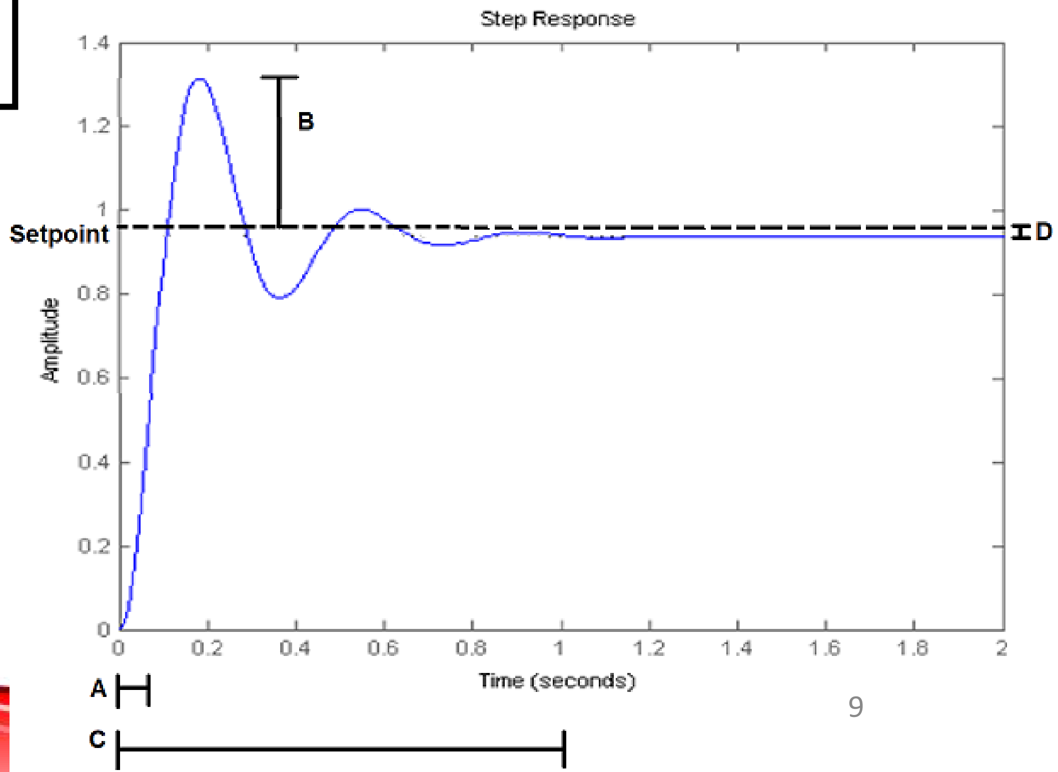


Feedback Control and Line Following

PID control

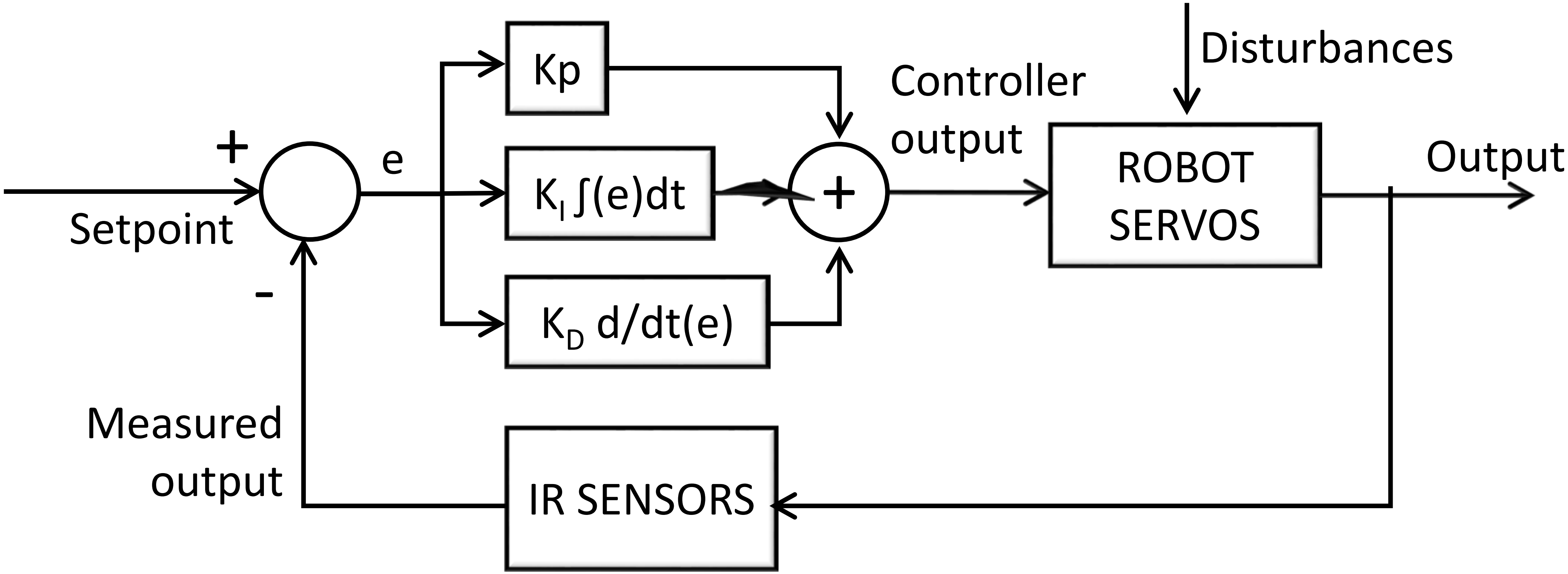


A - Rise Time
B - Percent Overshoot
C - Settling Time
D - Steady State Error



Feedback Control and Line Following

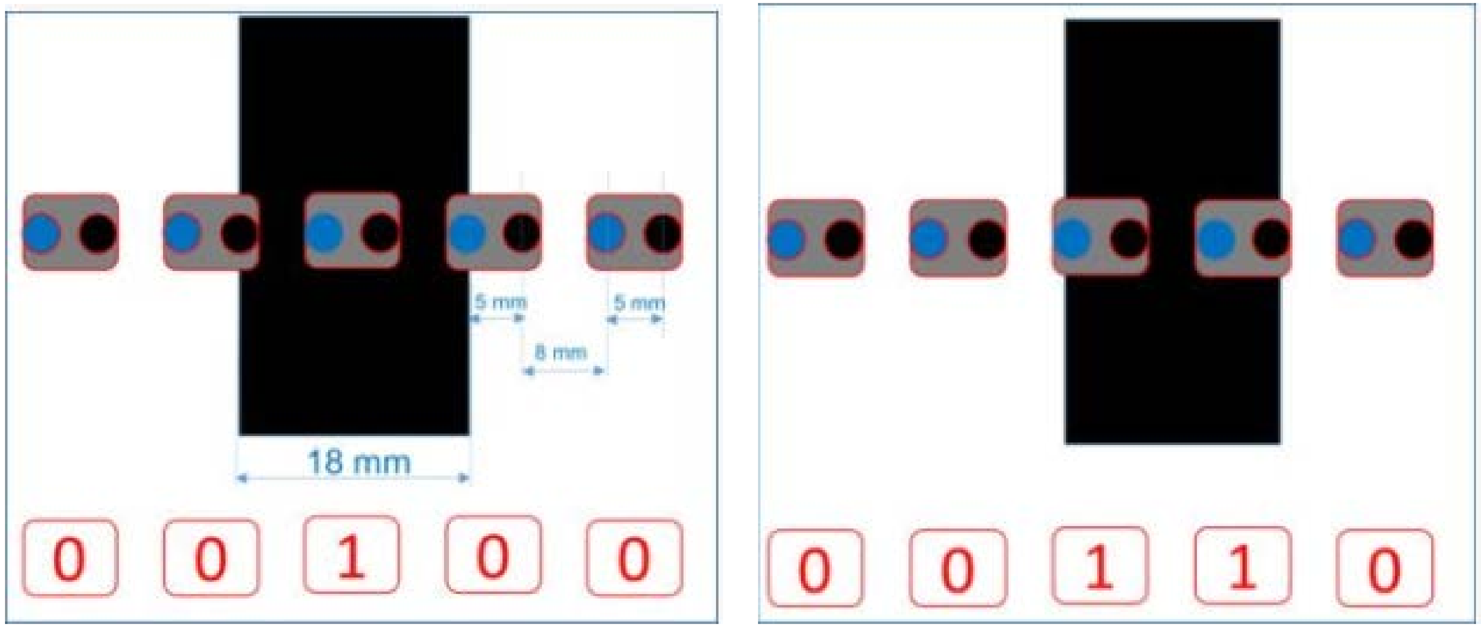
PID control



$$U(t) = K_P e(t) + K_I \int e(t)dt + K_D\, d/dt(e(t))$$

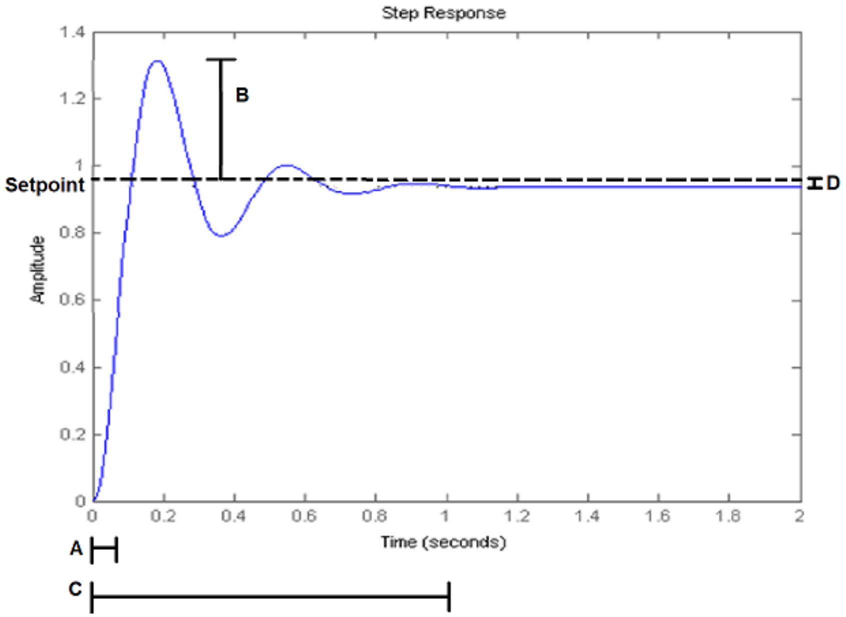
Feedback Control and Line Following

P control



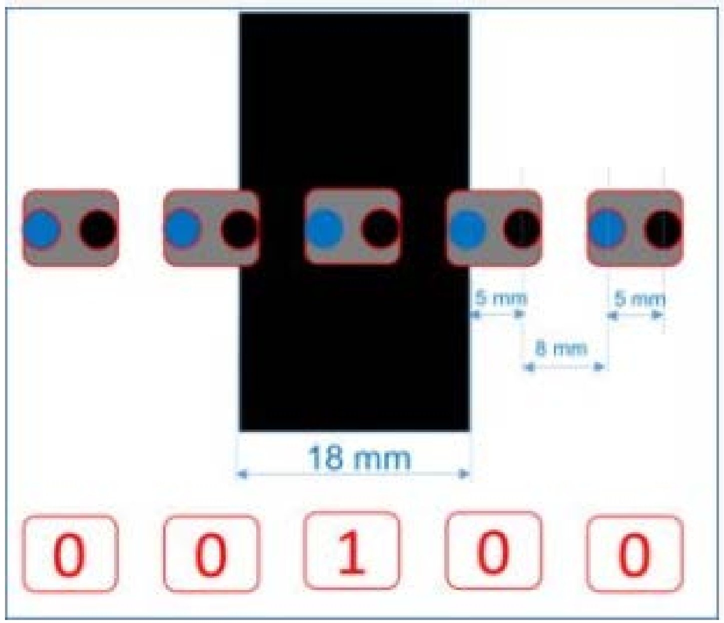

```
//P-controller:

void PControl(){
        error = IRmeasurements();
        motorSpeed = Kp*error + originalSpeed;
}
```

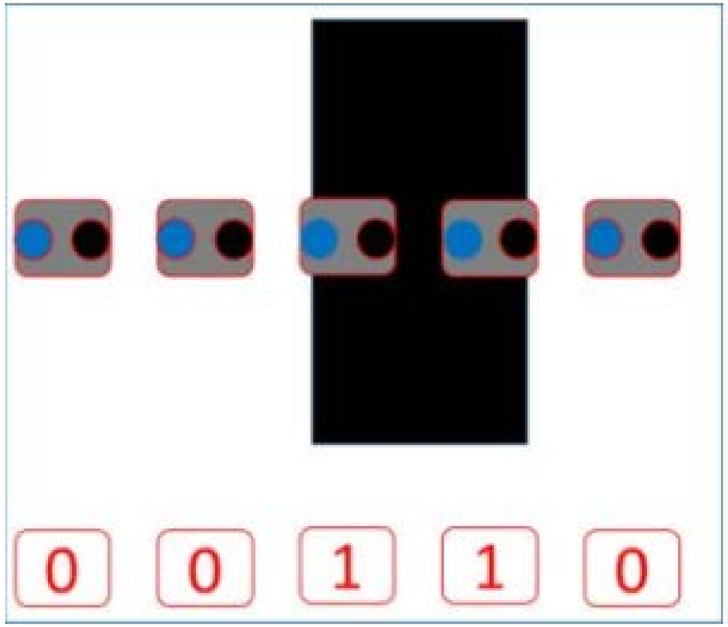
- *What's the disadvantage of PI control?*
  - Steady state error



Feedback Control and Line Following

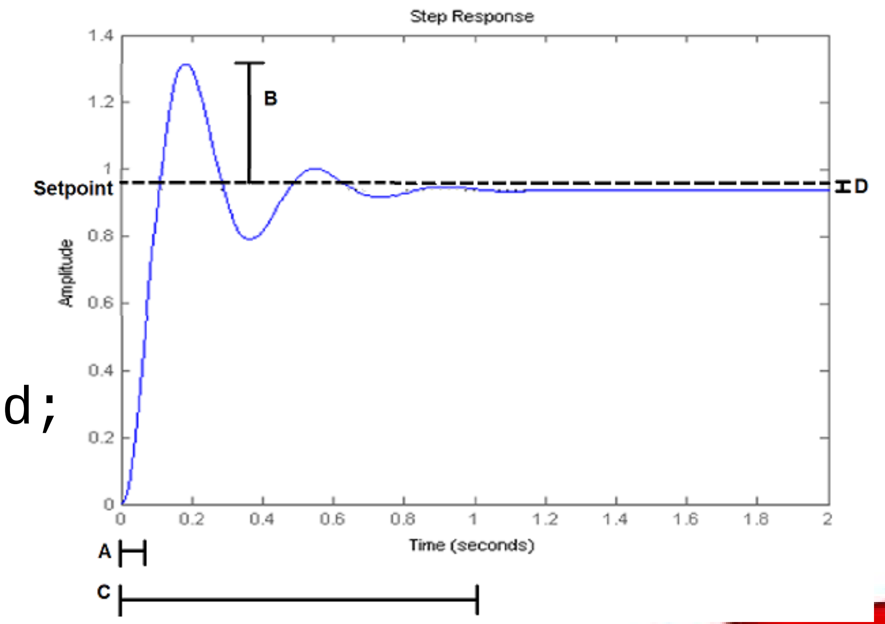
PI control





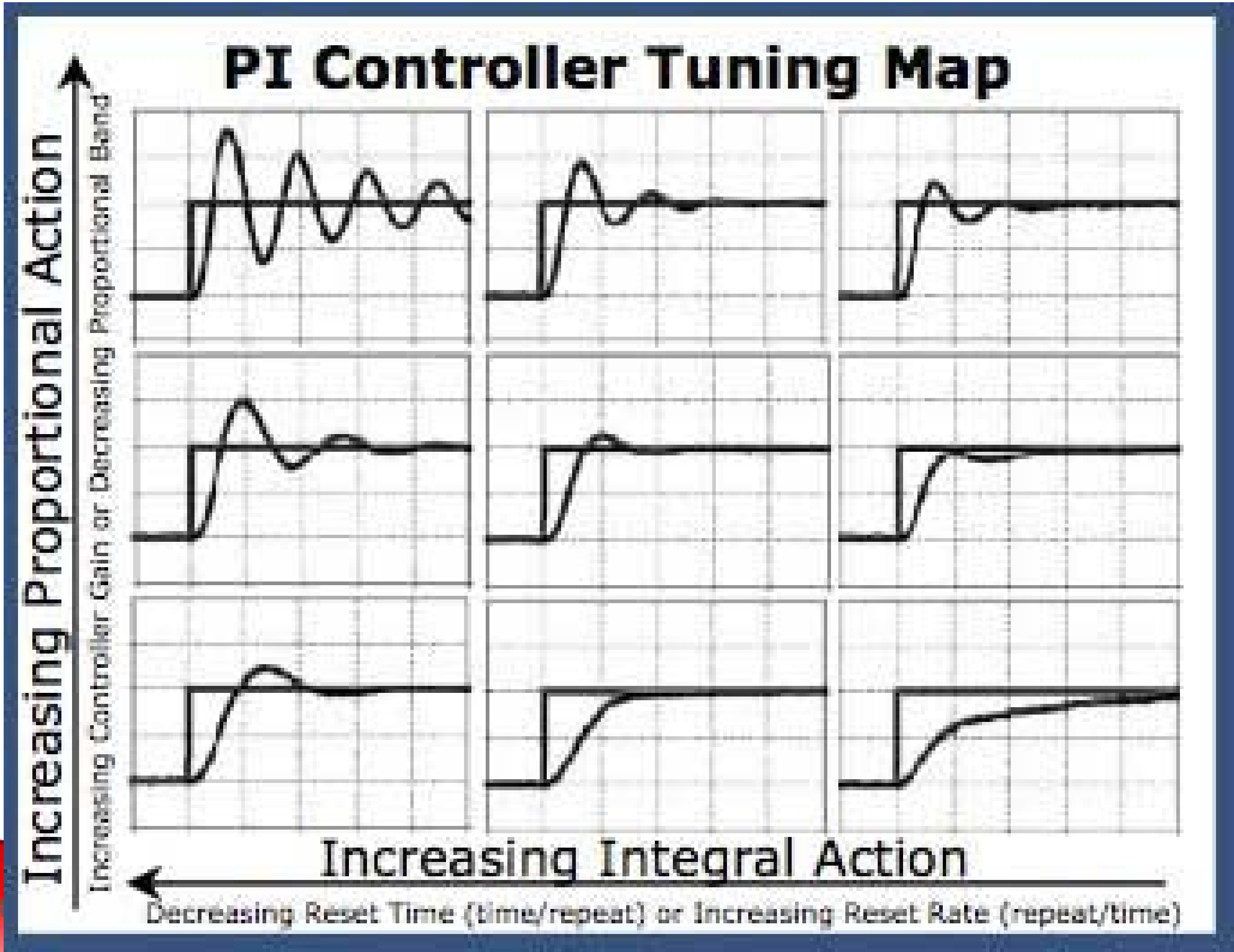
- *What's the disadvantage of PI control?*
  - Long settling time
  - More overshoot

```
//PI-controller:
void PIControl(){
      errorSum = errorSum + IRmeasurements()*dT;
      motorSpeed =
            Kp*error + Ki+errorSum + originalSpeed;
}
```



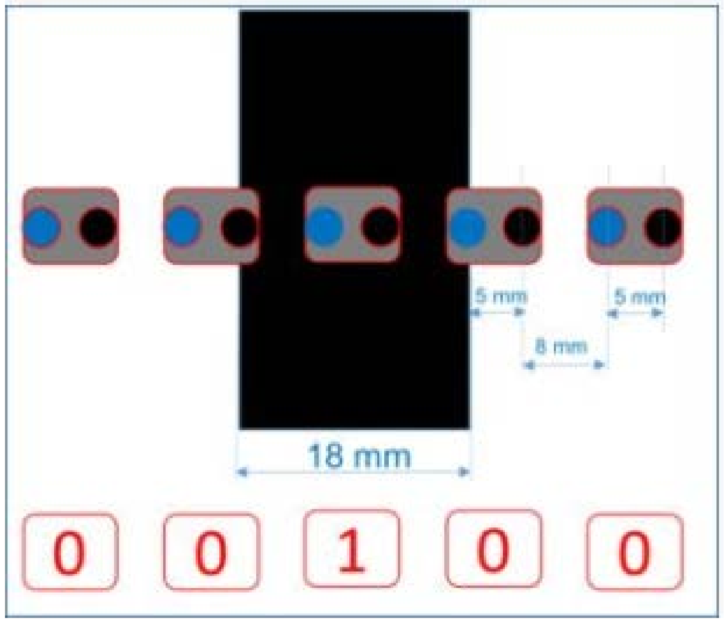
Rise time, settling time, and overshoot

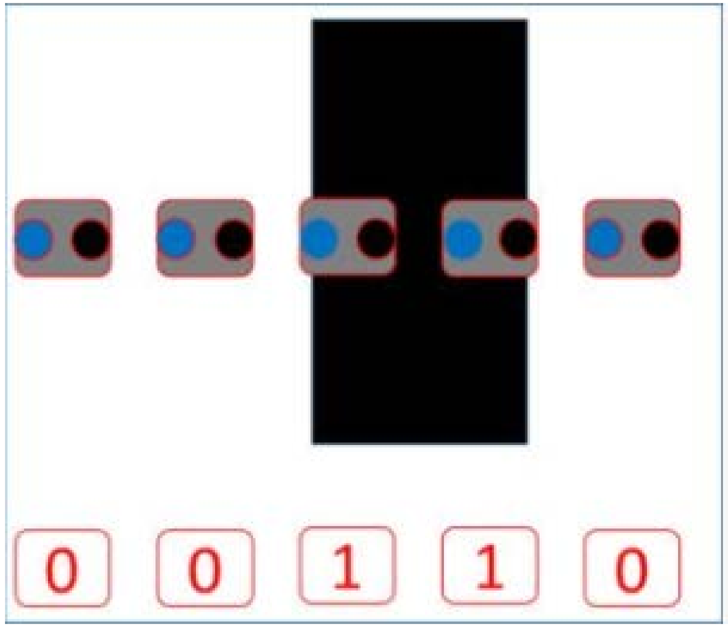
- *Where would you want your system to be?*



Feedback Control and Line Following

PID control





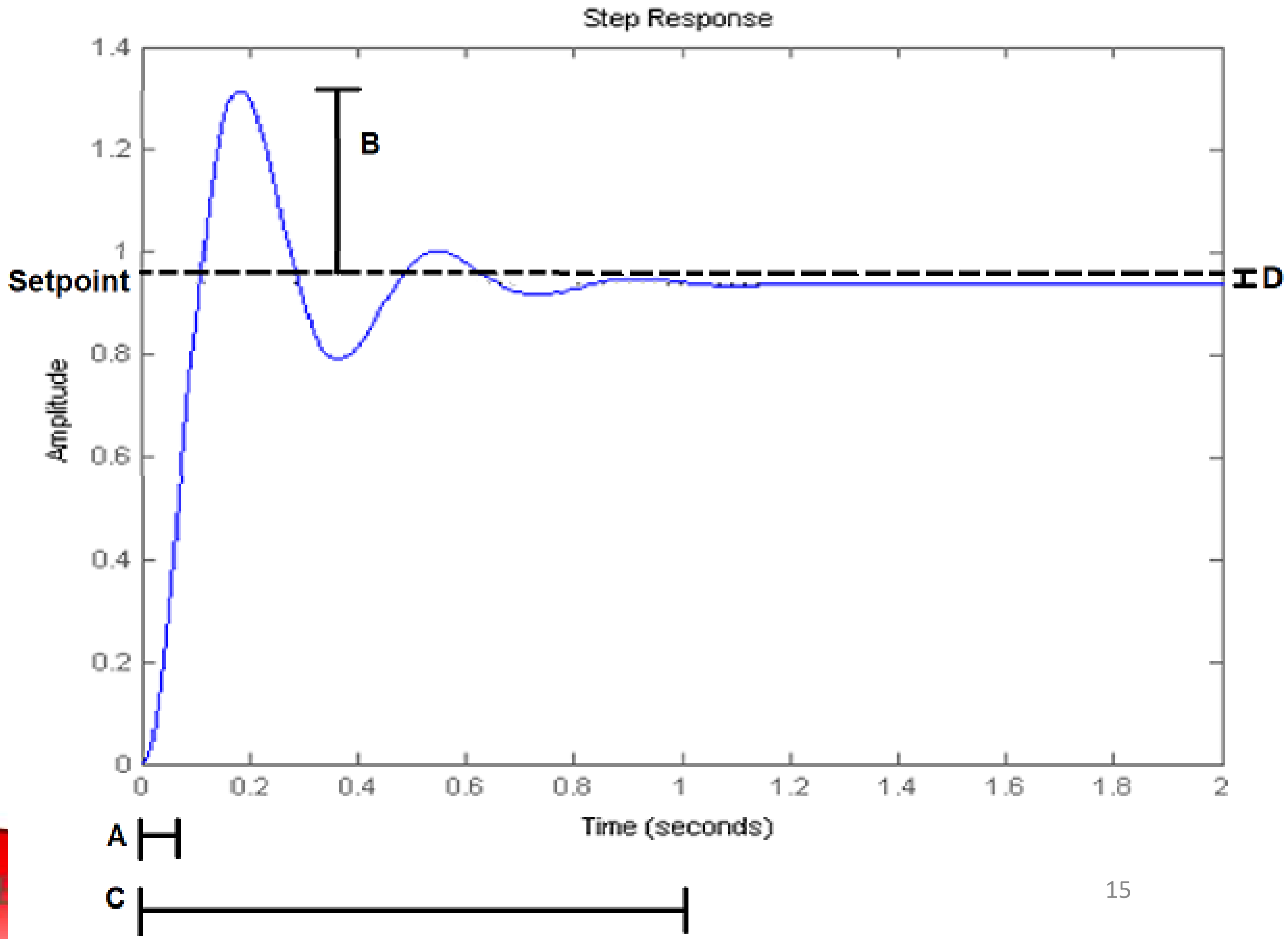

```
//PID-controller:
void PIDControl(){
      errorSum = errorSum + IRmeasurements()*dT;
      errorDiff = (error – lastError)/dT;
      motorSpeed =
            Kp*error + Ki+errorSum +
            Kd*errorDiff + originalSpeed;
}
```

Feedback Control and Line Following

PID control

Pointers for adjusting your control:

- You can decrease t_{rise} and e_{ss} by increasing K_P, at the expense of increased overshoot
- You can decrease t_{rise} and eliminate e_{ss} by increasing K_I at the expense of increased overshoot and settling time
- You can decrease the overshoot and the settling time by increasing K_d

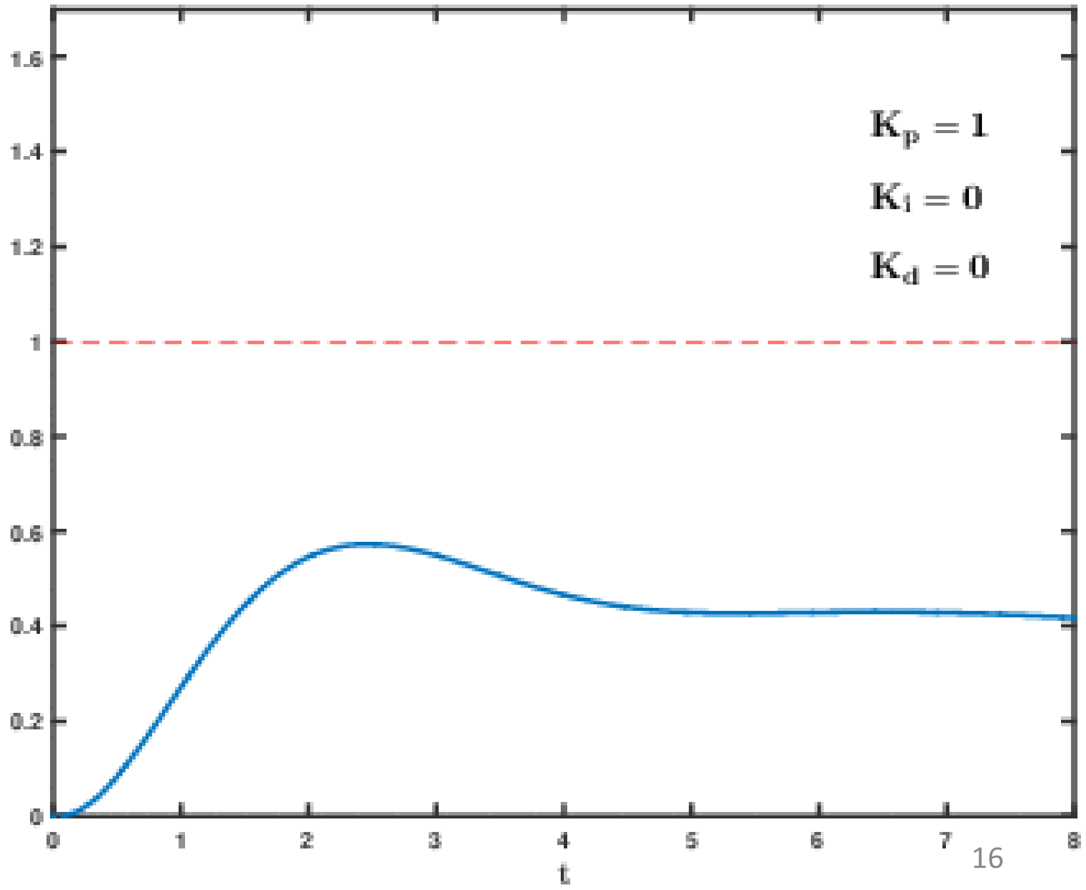


Feedback Control and Line Following

PID control

Pointers for adjusting your control:

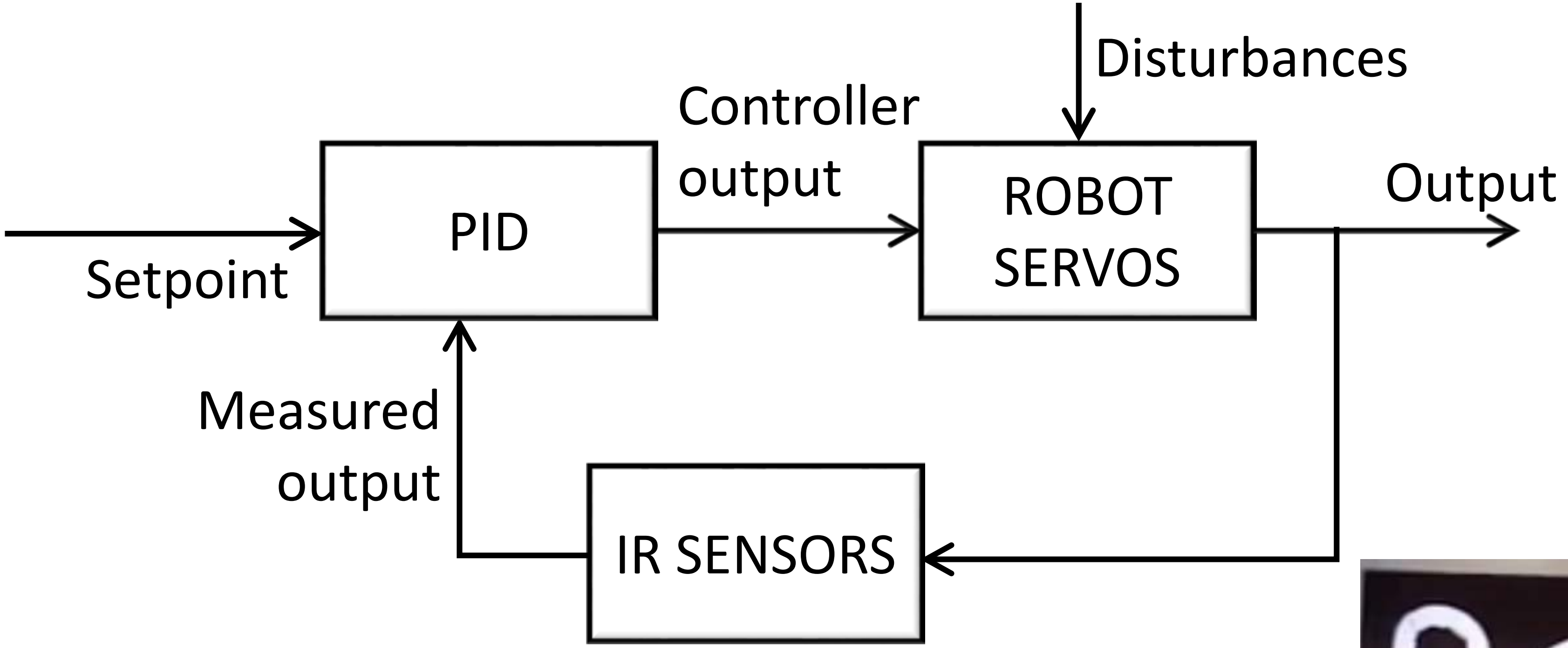
- Set all coefficients to zero
- Increase Kp until system oscillates
- Increase Ki until steady state error corrected
- Increase Kd until overshoot decreased



ECE3400 Cornell Engineering Electrical and Computer Engineering

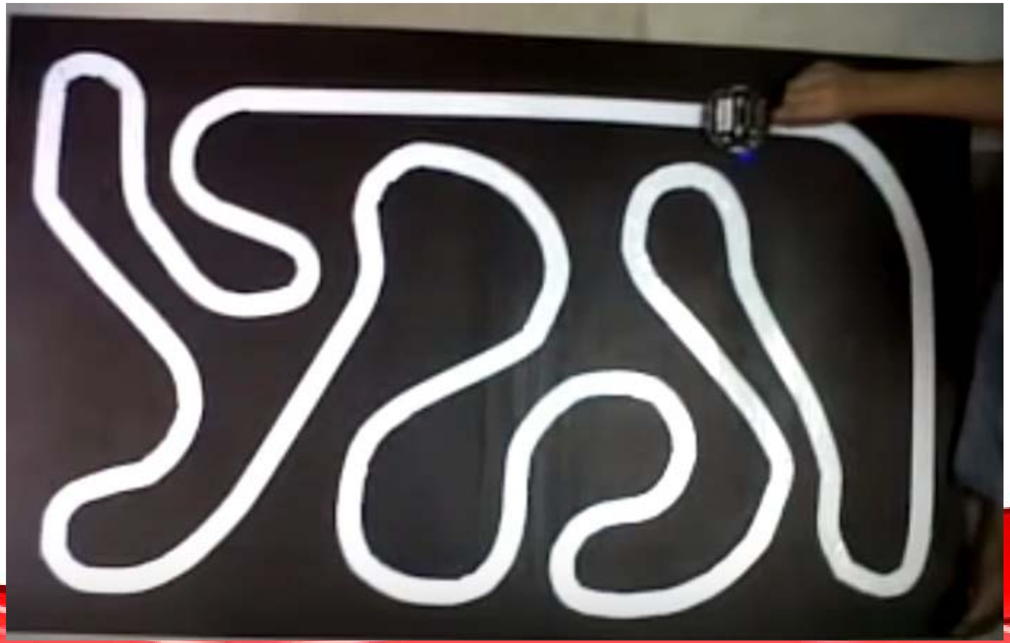
Feedback Control and Line Following

PID control



- *Proportional:* Looks at the instantaneous error
- *Integral:* Sum of errors over time
- *Derivative:* Rate of change of error over time

www.Polulu.com

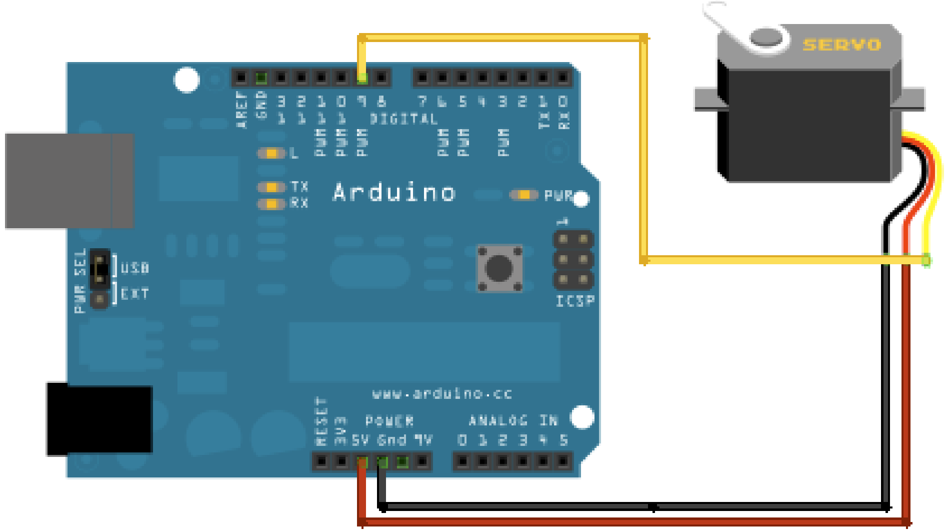


Have you ever wondered what is inside your servo?

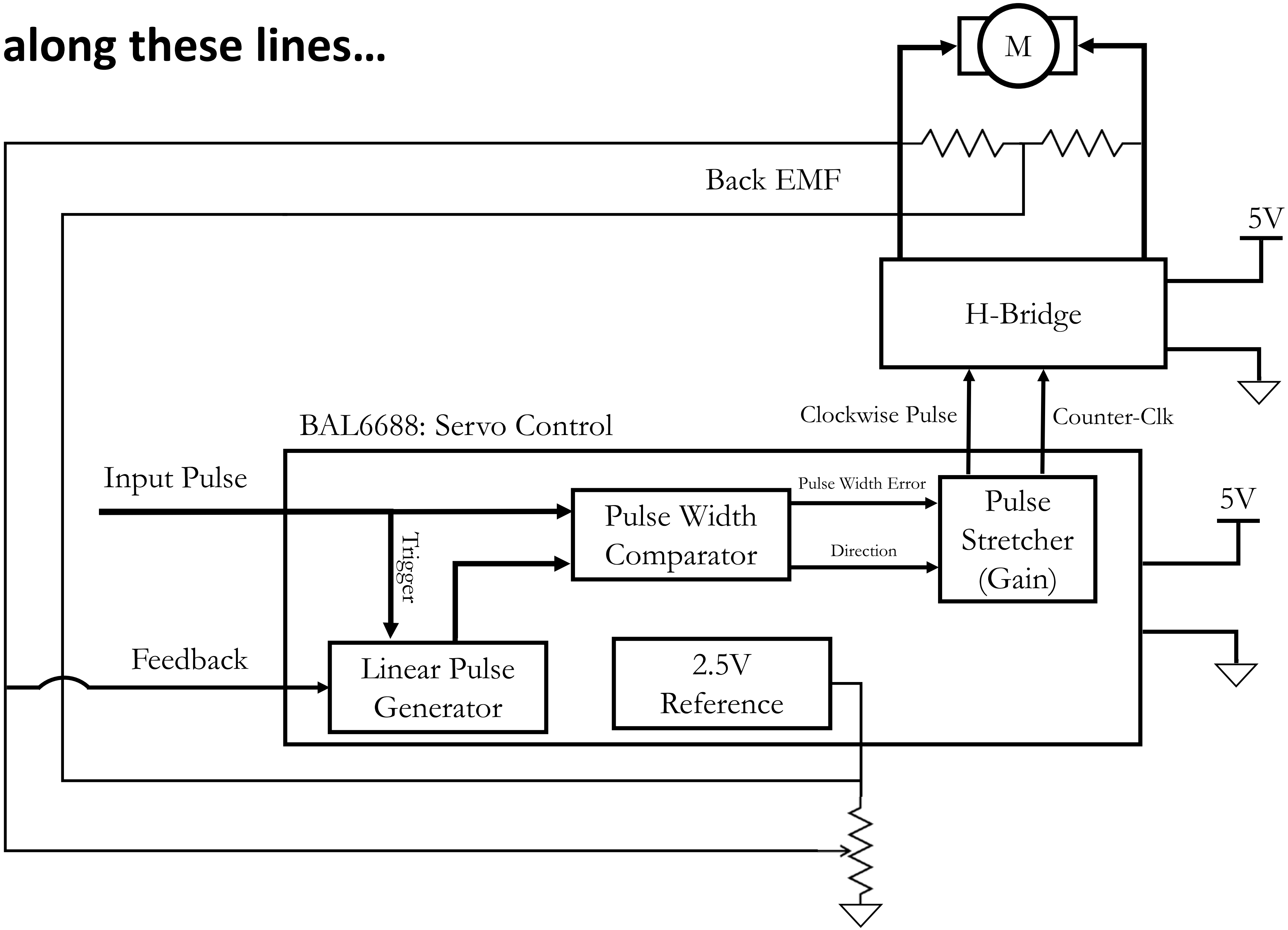
Positional | Continuous

Have you ever wondered what's inside your servo?

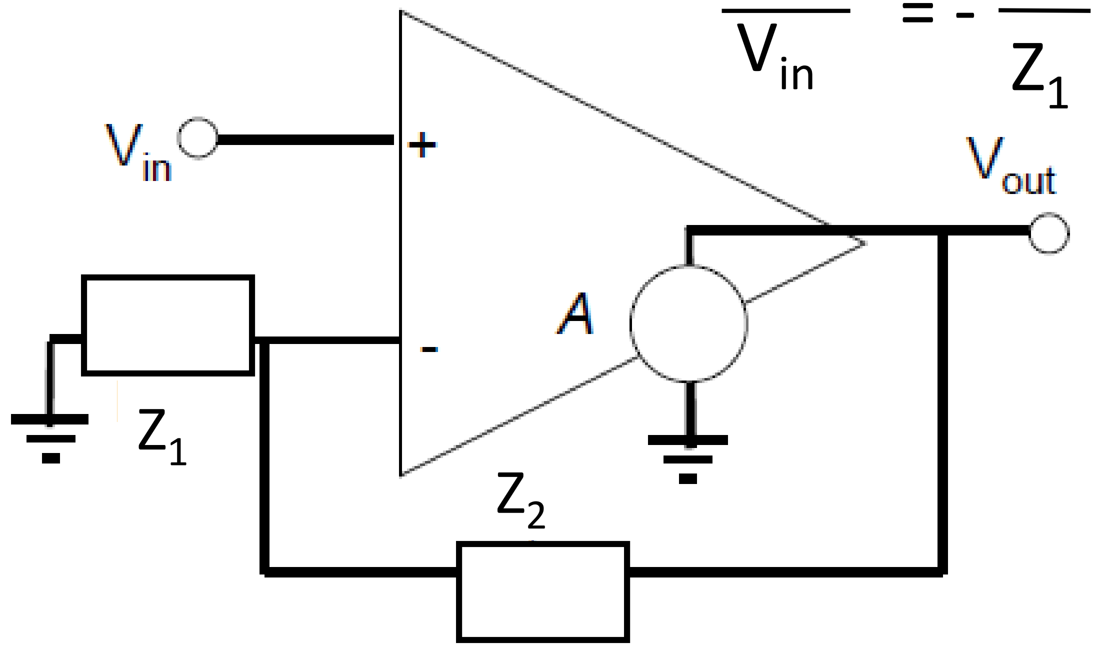
- *What are the two standard types of servos?*
  - Continuous
  - Positional
- *How do we get positional control?*
  - Potentiometer
- *How can you change a positional control to a speed control?*
  - Cut the feedback loop!
- *How do we get speed control?*
  - Back EMF



Something along these lines…



Analog Feedback Control



$$\frac{V_{out}}{V_{in}} = -\frac{Z_2}{Z_1}$$

Function	Z_1	Z_2	$H(s)$
Gain (P)	R_1	R_2	$-R_2/R_1$
Integration (I)	R	C	$-(RC)^{-1}/s$
Differentiation (D)	C	R	$-RCs$
PI control	R_1	R_2C	$\frac{-R_2(s+(R_2C)^{-1})}{R_1 s}$
PD control	$C\|\|R_1$	R_2	$-R_2C\,(s+(R_1C)^{-1})$
PID control	$C_1\|\|R_1$	R_2C_2	

Go Build Robots!