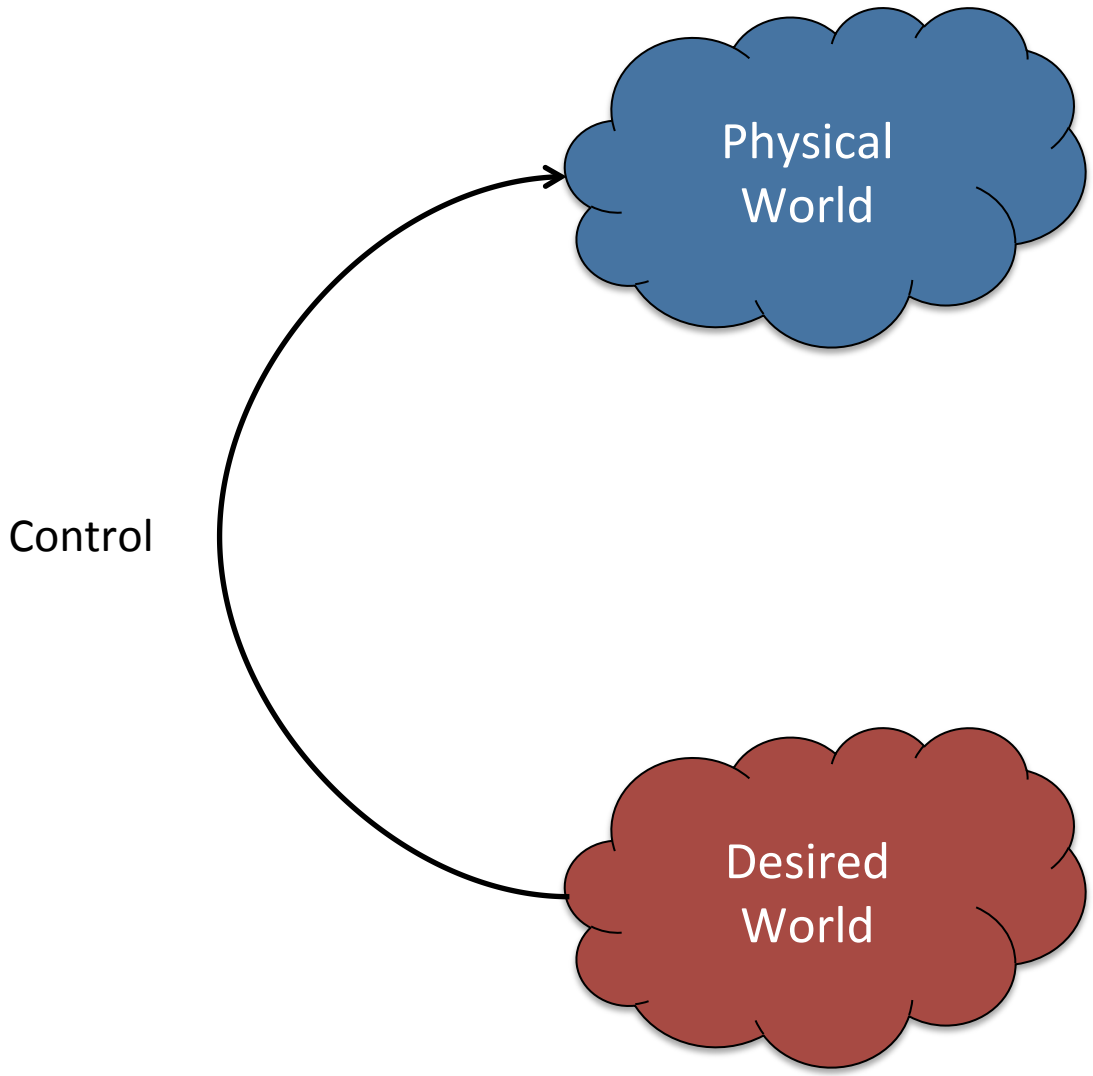
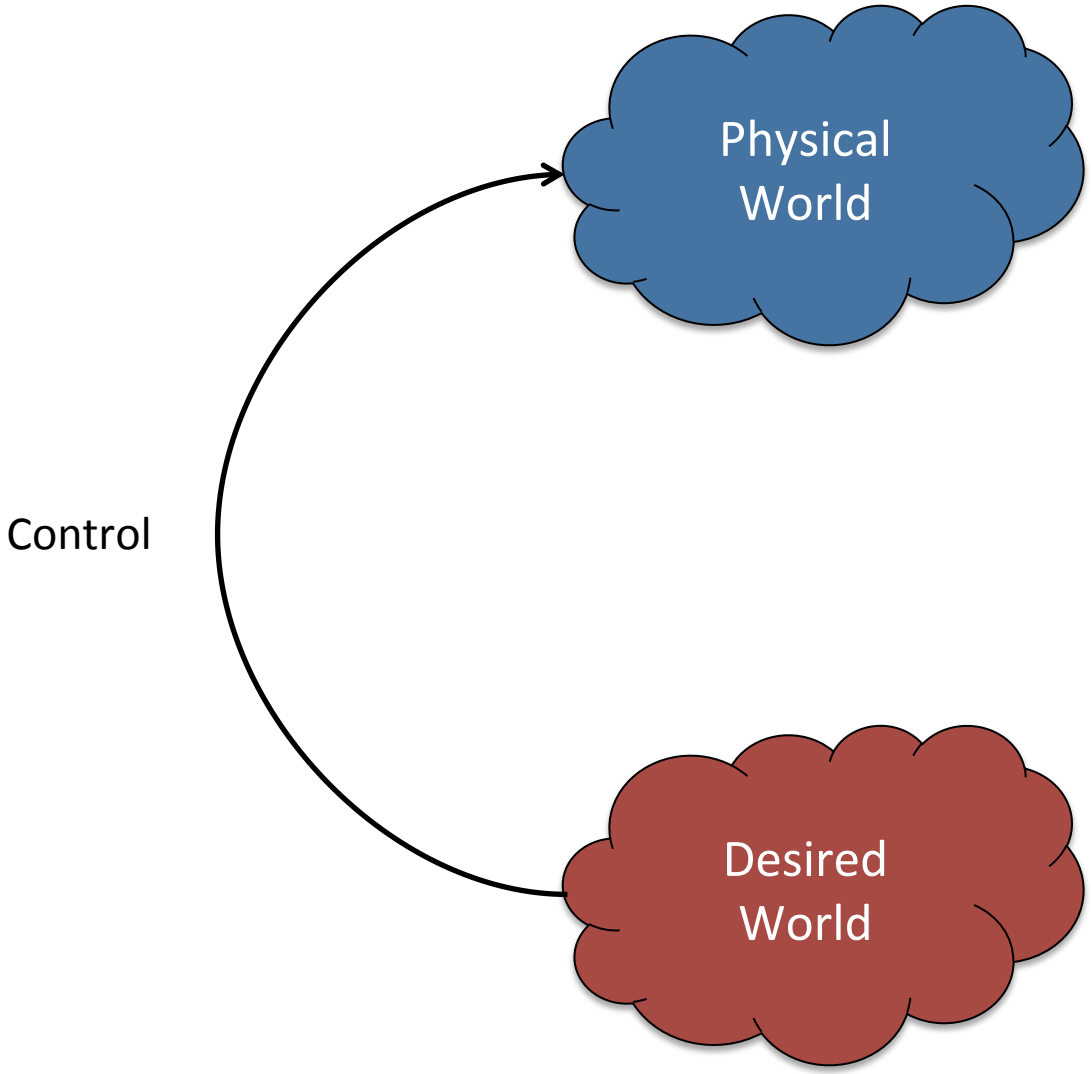
PID

Sensing and Control Systems

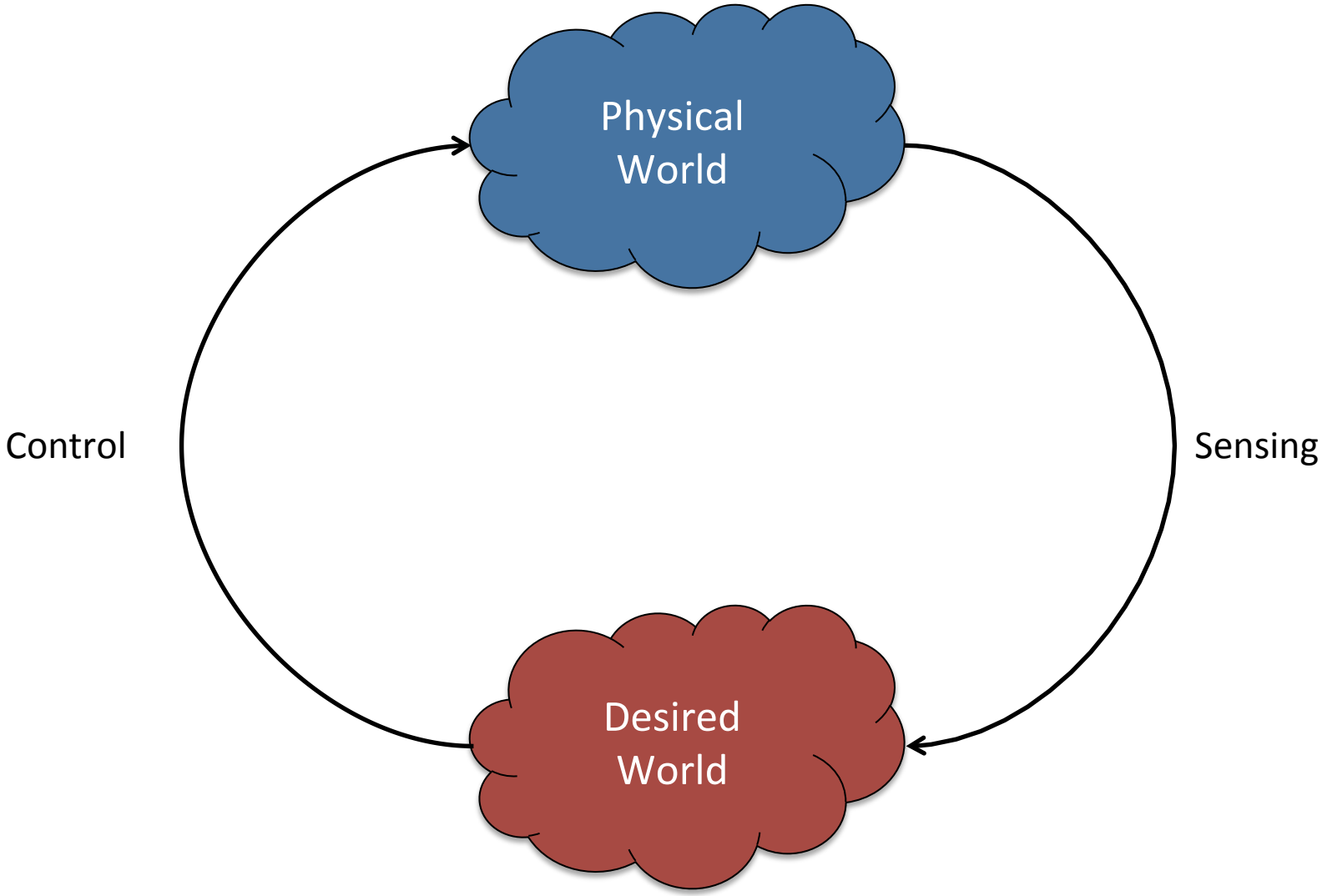
Ara Kourchians


Physical
World
Control
Desired
World

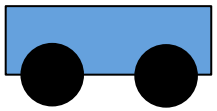
Open Loop

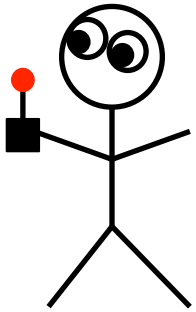


Closed Loop

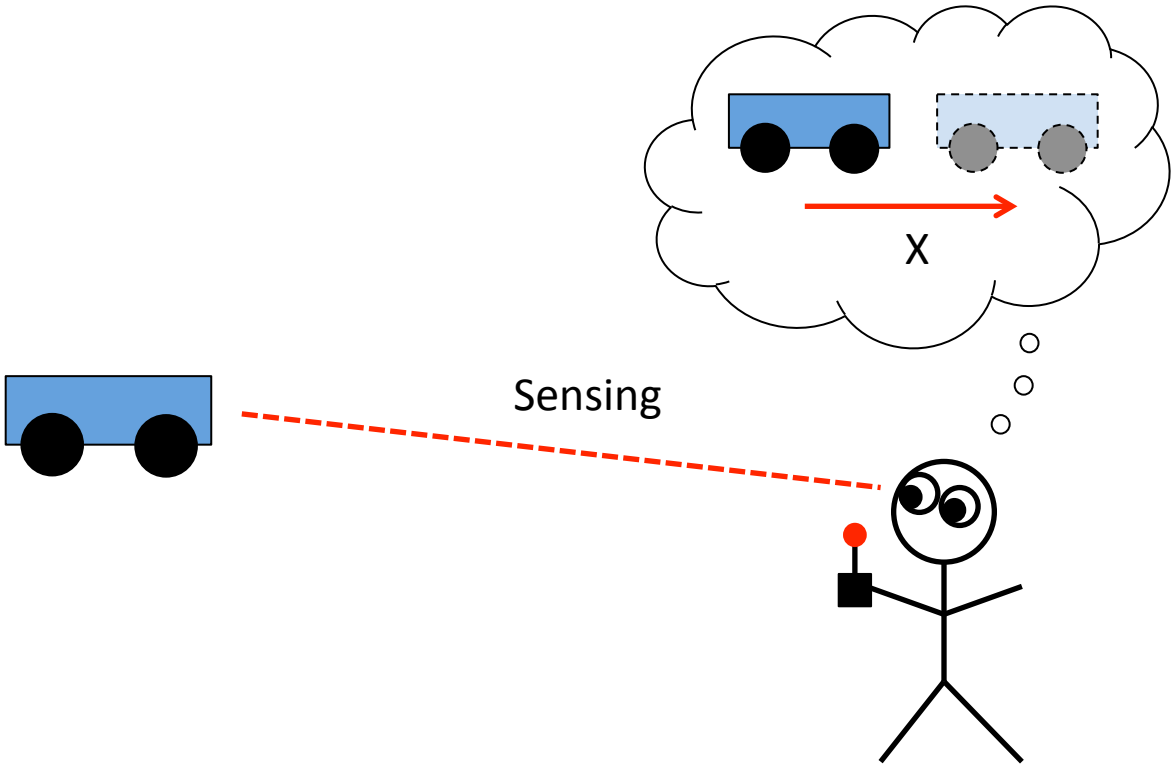


??? Loop

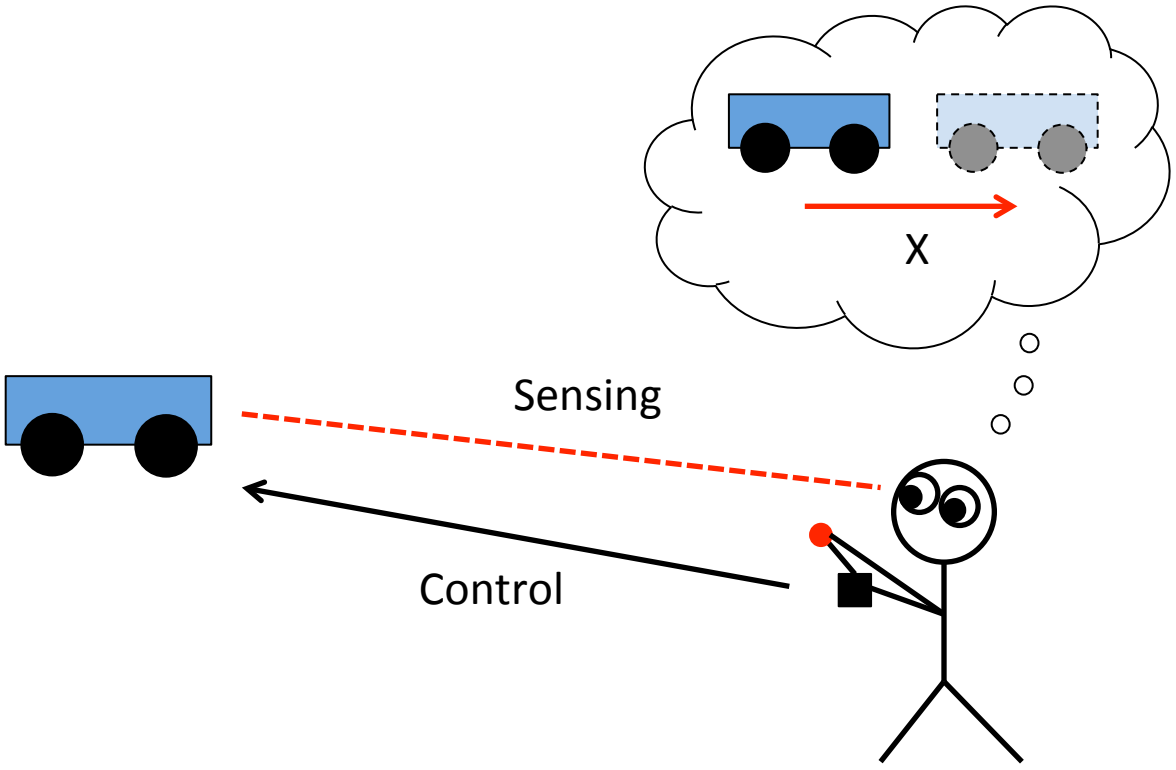




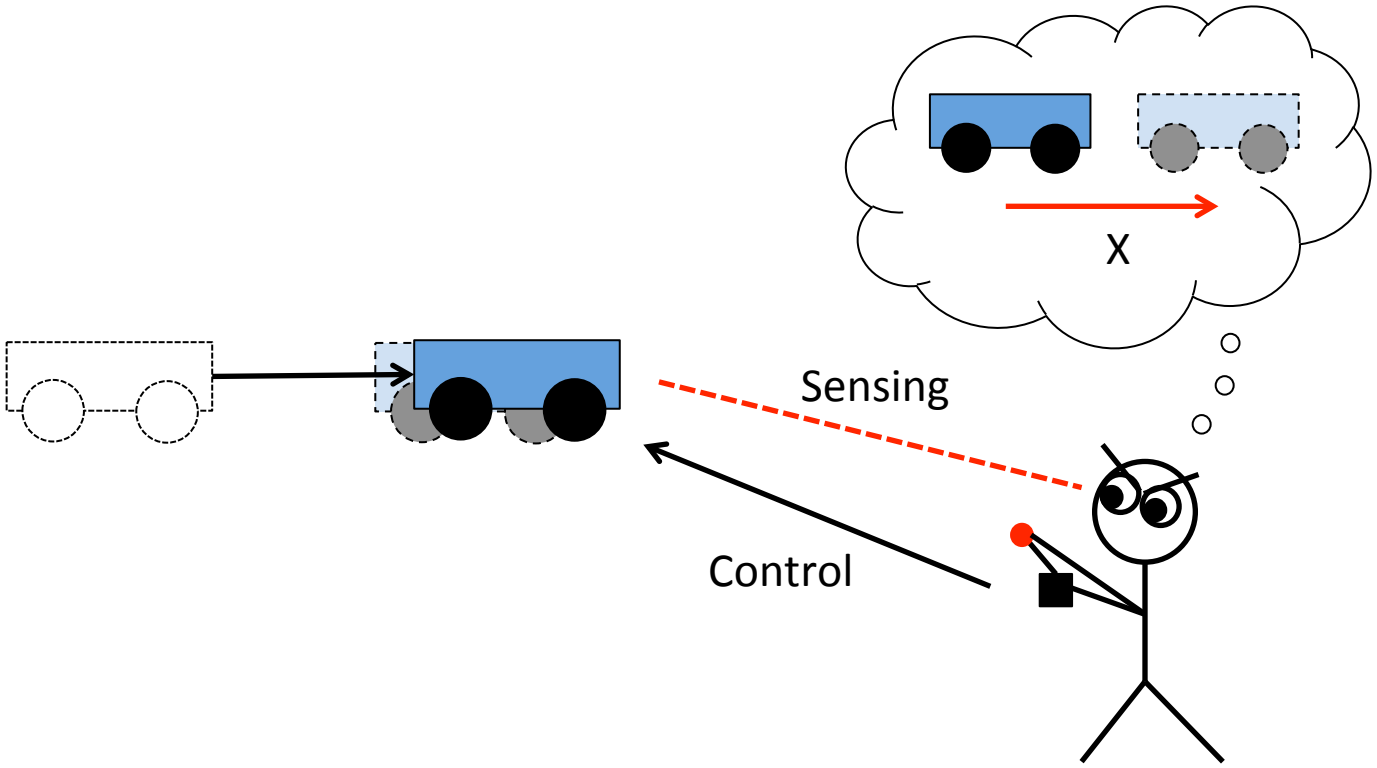
Closed Loop



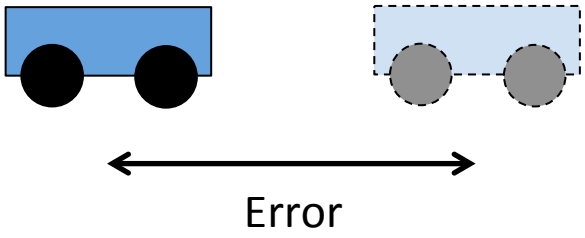
Closed Loop



Closed Loop

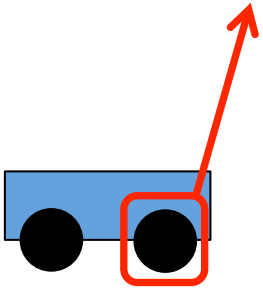


Some terminology

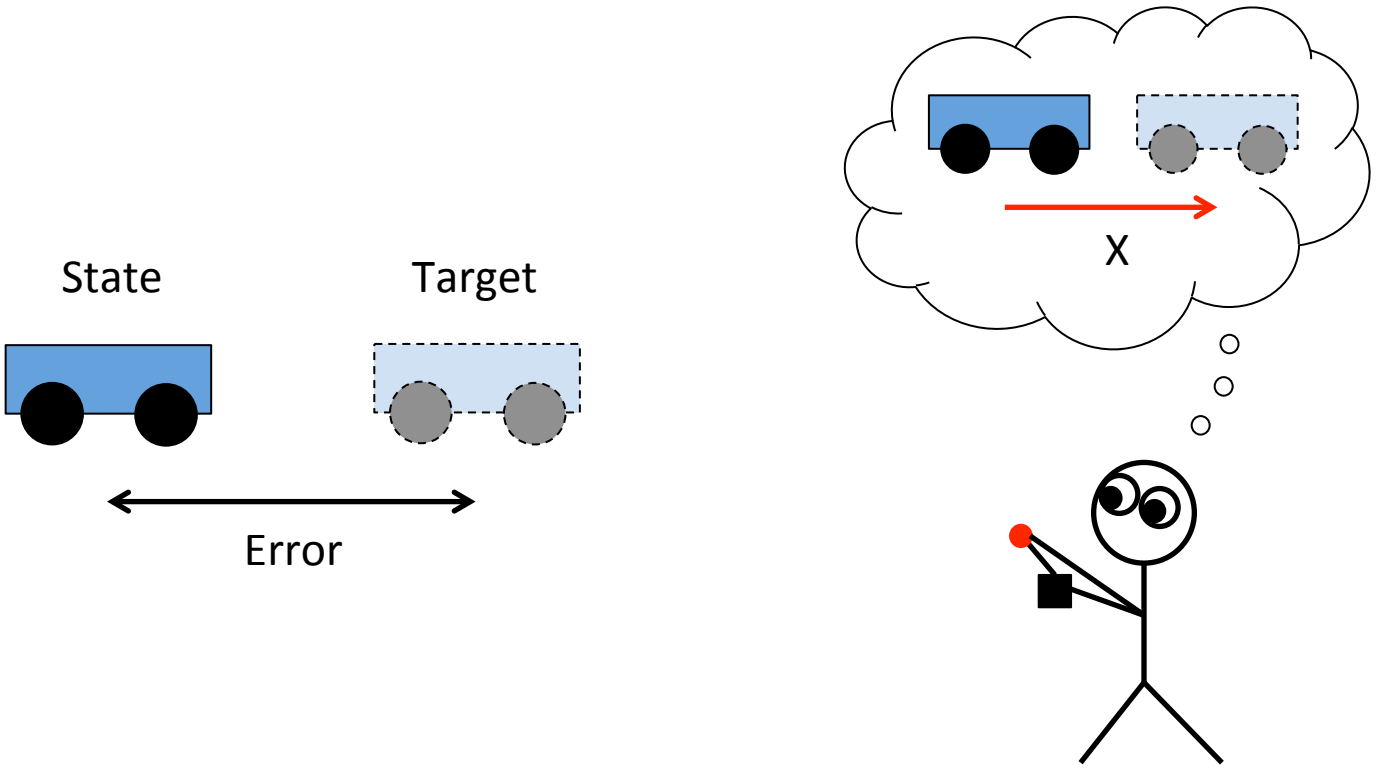


Motor

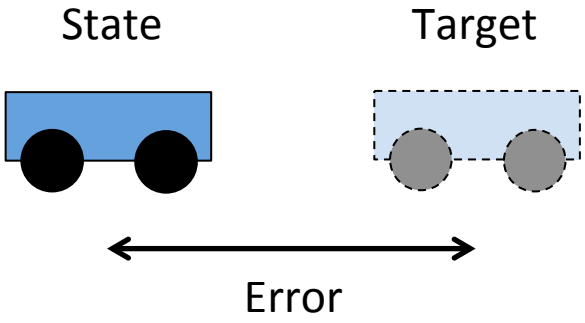
Encoder

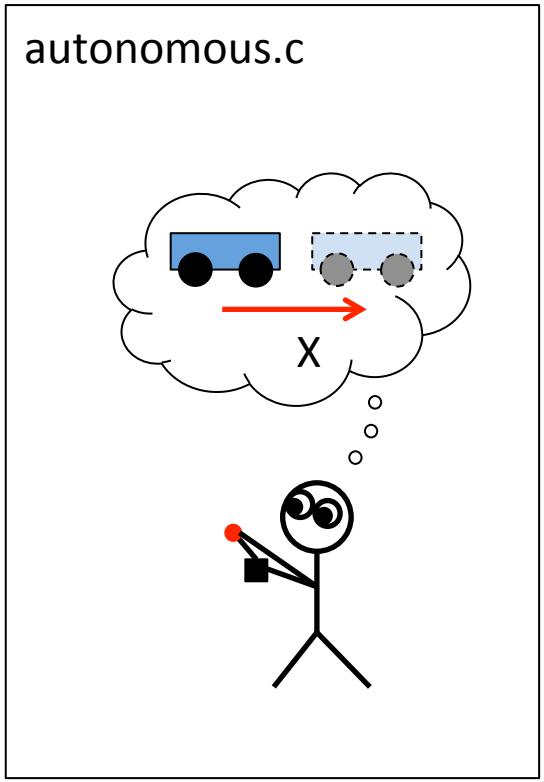


Autonomous Mode

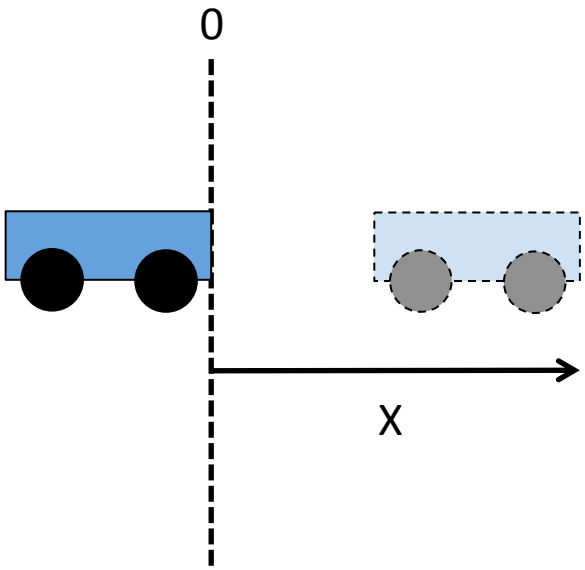


Autonomous Mode





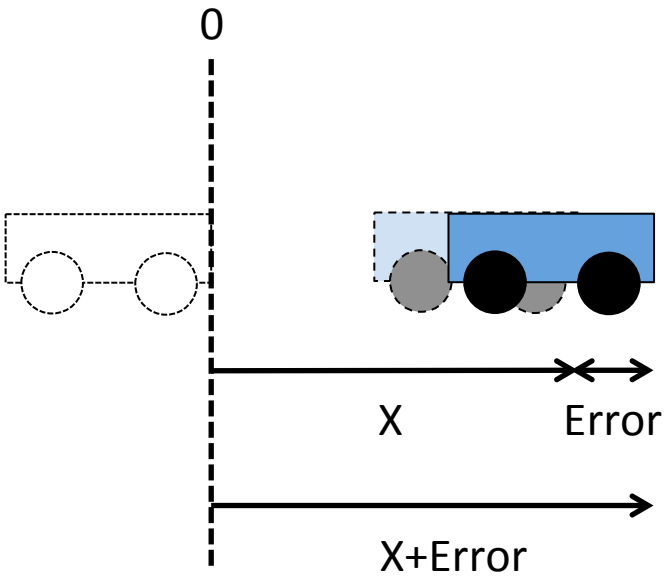
Autonomous Mode



autonomous.c

```
void go()
{
        ..
        robot.drive(X);
        ..
}
```

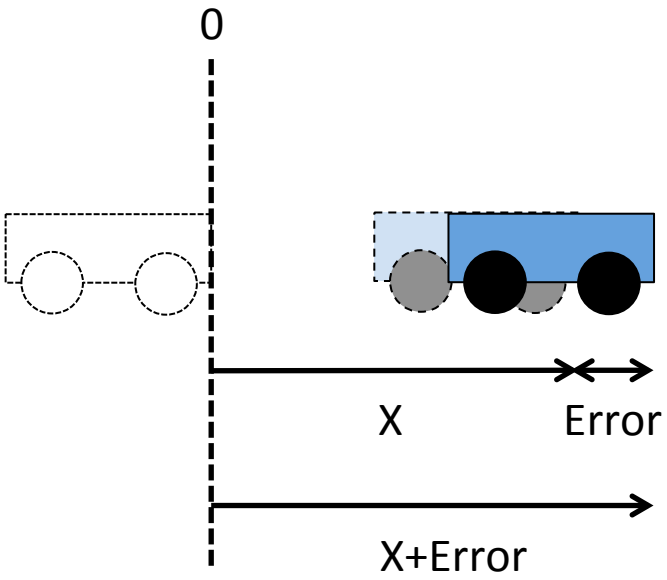
Autonomous Mode



autonomous.c

```
void go()
{
        ..
        robot.motors(100);
        wait(1_SEC);
        robot.motors(0);
        ..
}
```

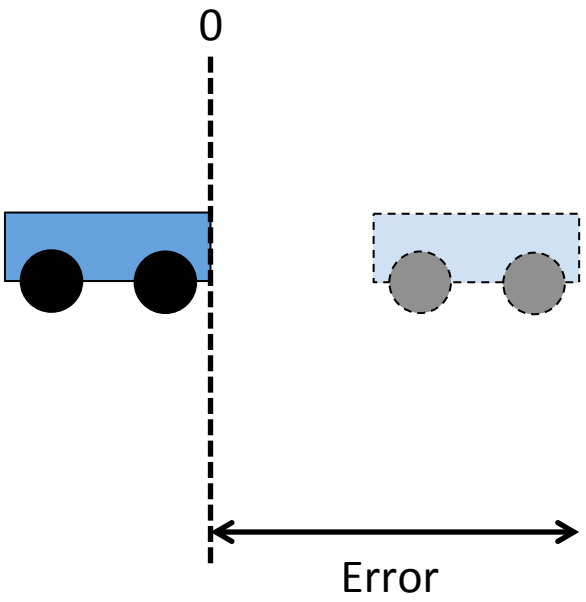
Autonomous Mode



autonomous.c

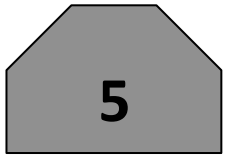
```
void go()
{
        ..
        robot.motors(100);

        while(encoder < 1000)
        {
          encoder = robot.read();
        }

        robot.motors(0);
        ..
}
```

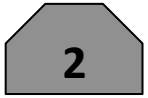
PID Control

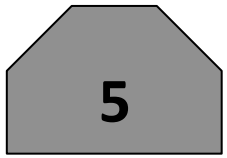


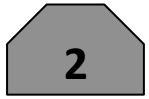
PID Control

Proportional		**I**ntegral		**D**erivative
Present		Past		Future
P*Error	**+**	**I***Sum_of_Errors	**+**	**D***(Error- Last_Error)



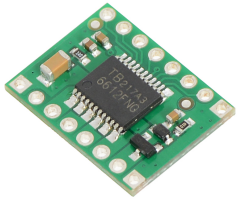


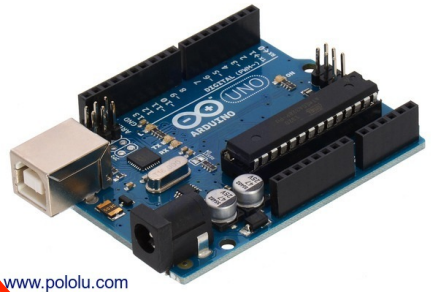


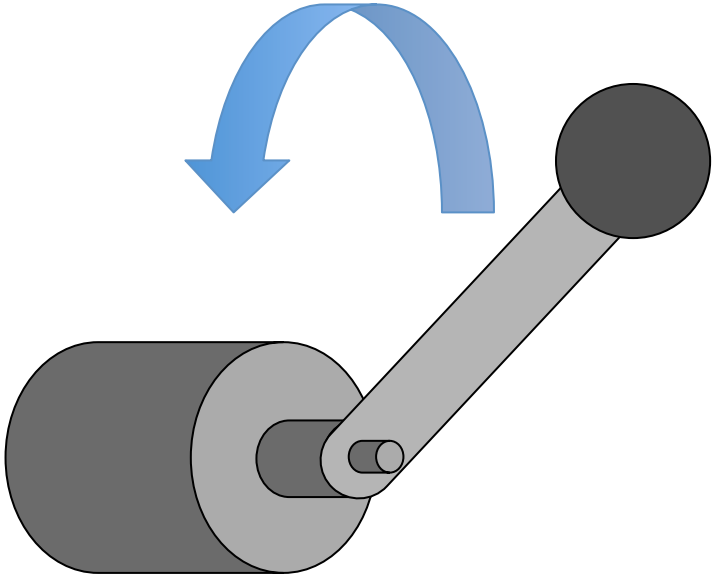


PID in the real world

Next Time…



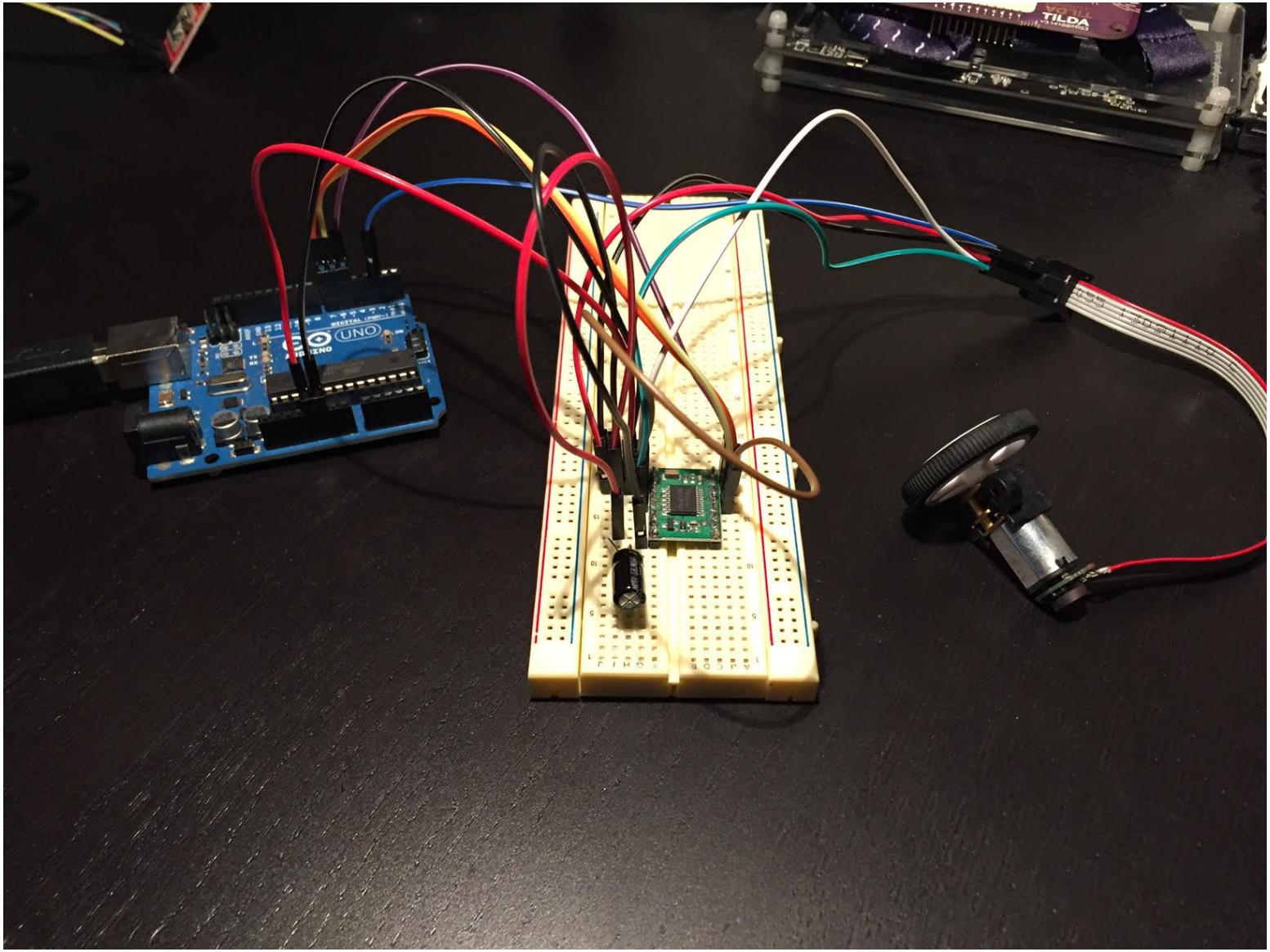




Hardware:
October 1st

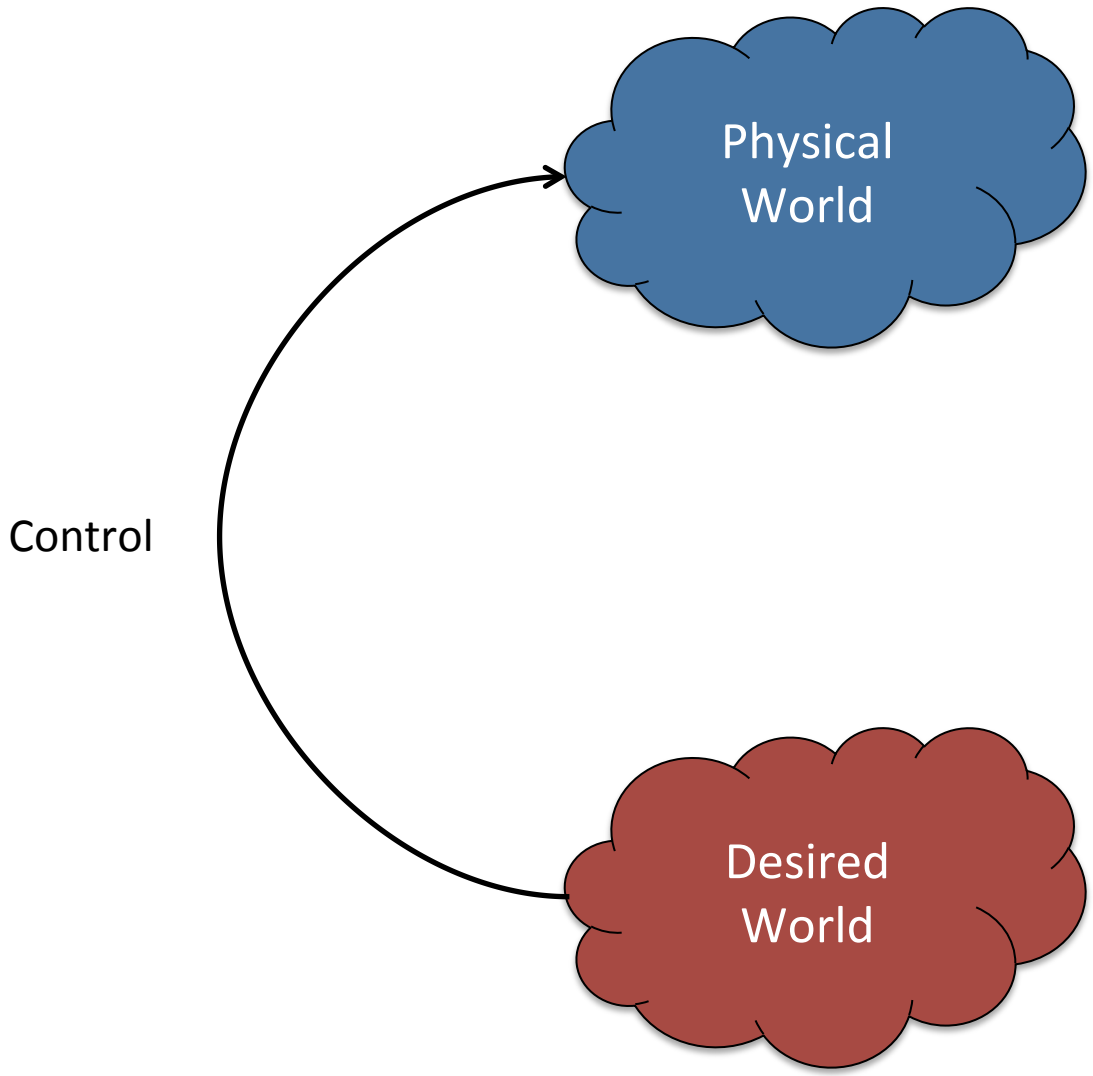
Math and Code:
October 8th

Next Time…

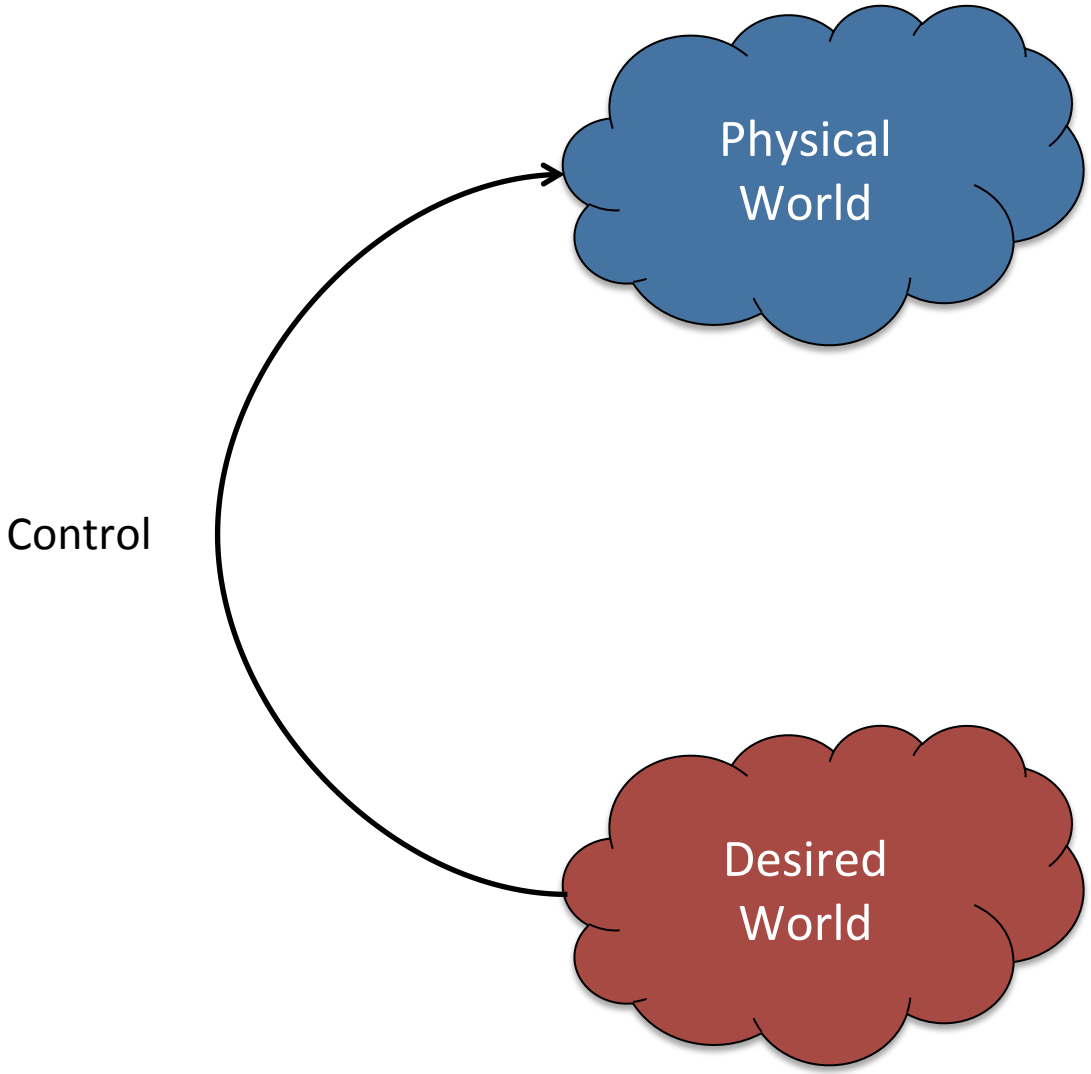


PID

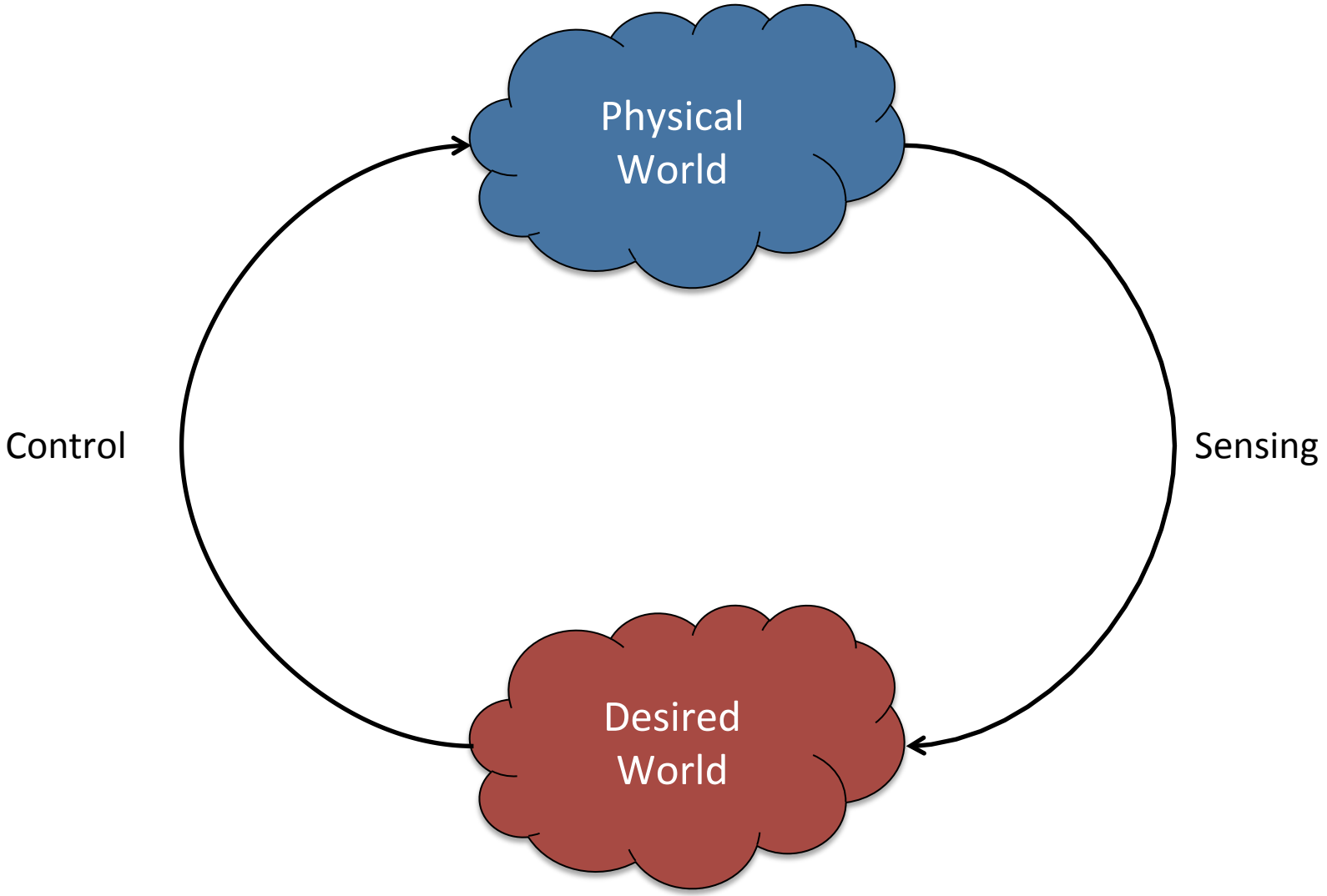
Ara Kourchians


Physical
World
Control
Desired
World

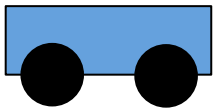
Open Loop

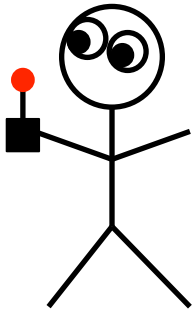


Closed Loop

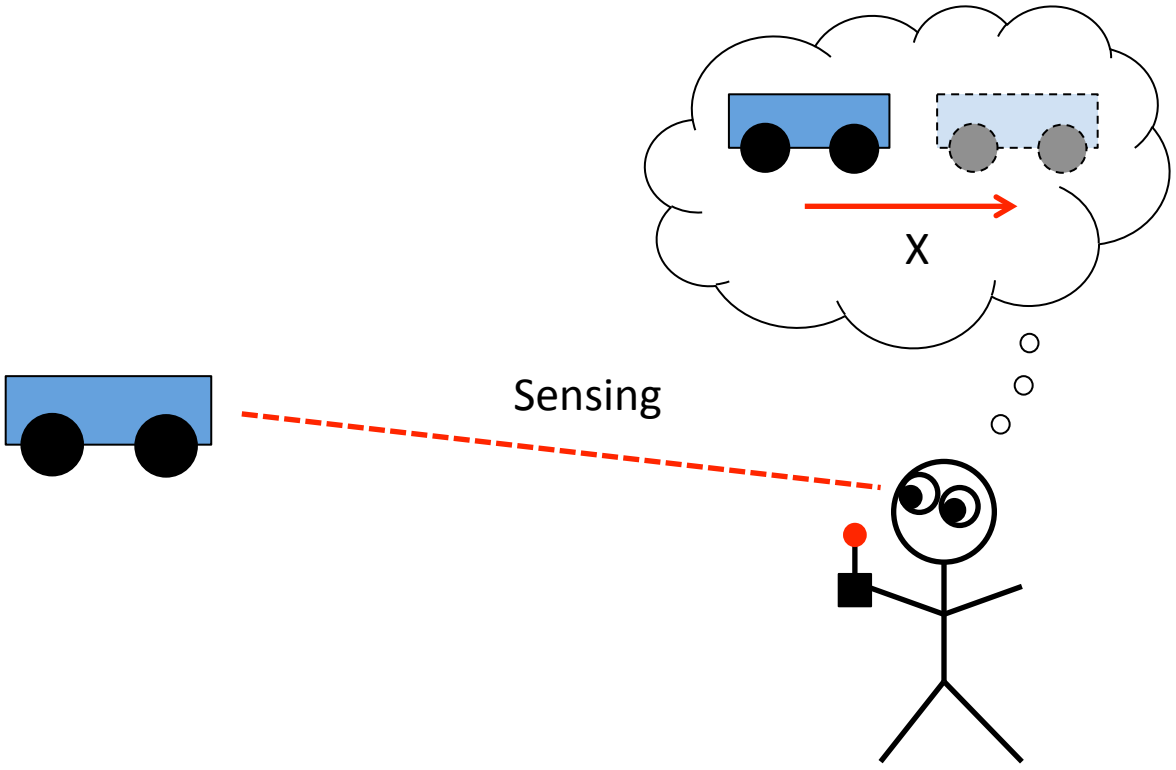


??? Loop

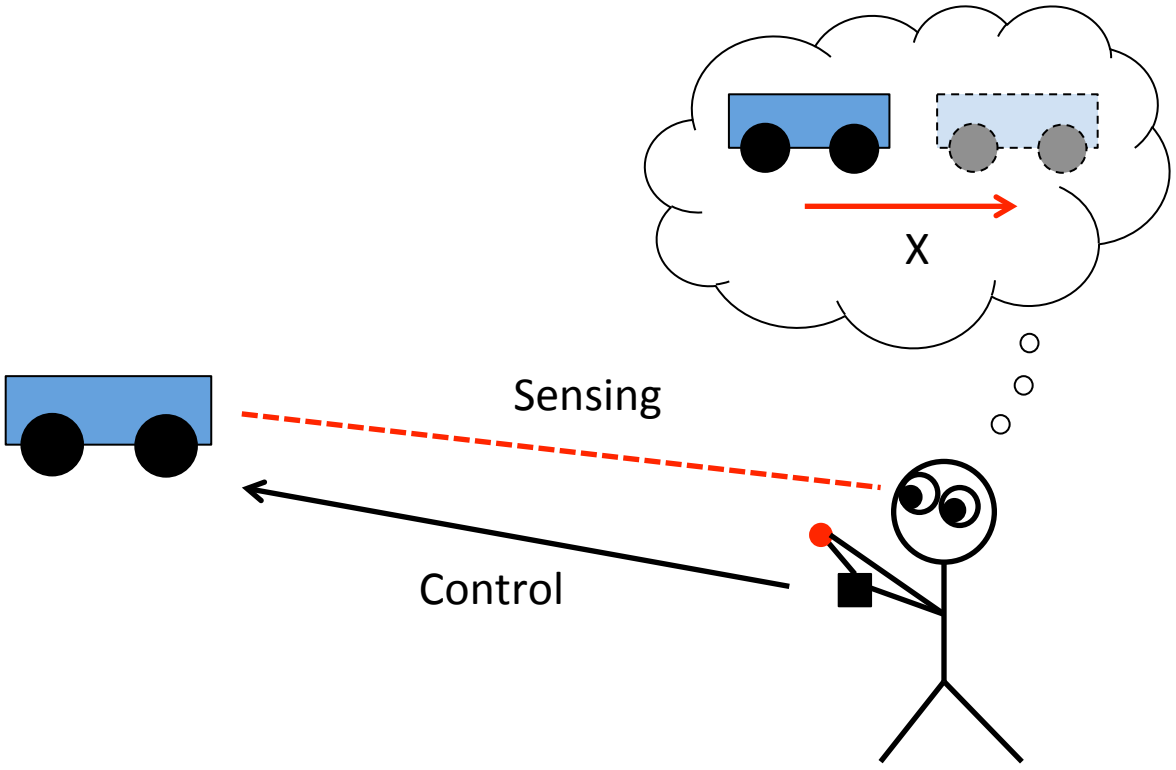




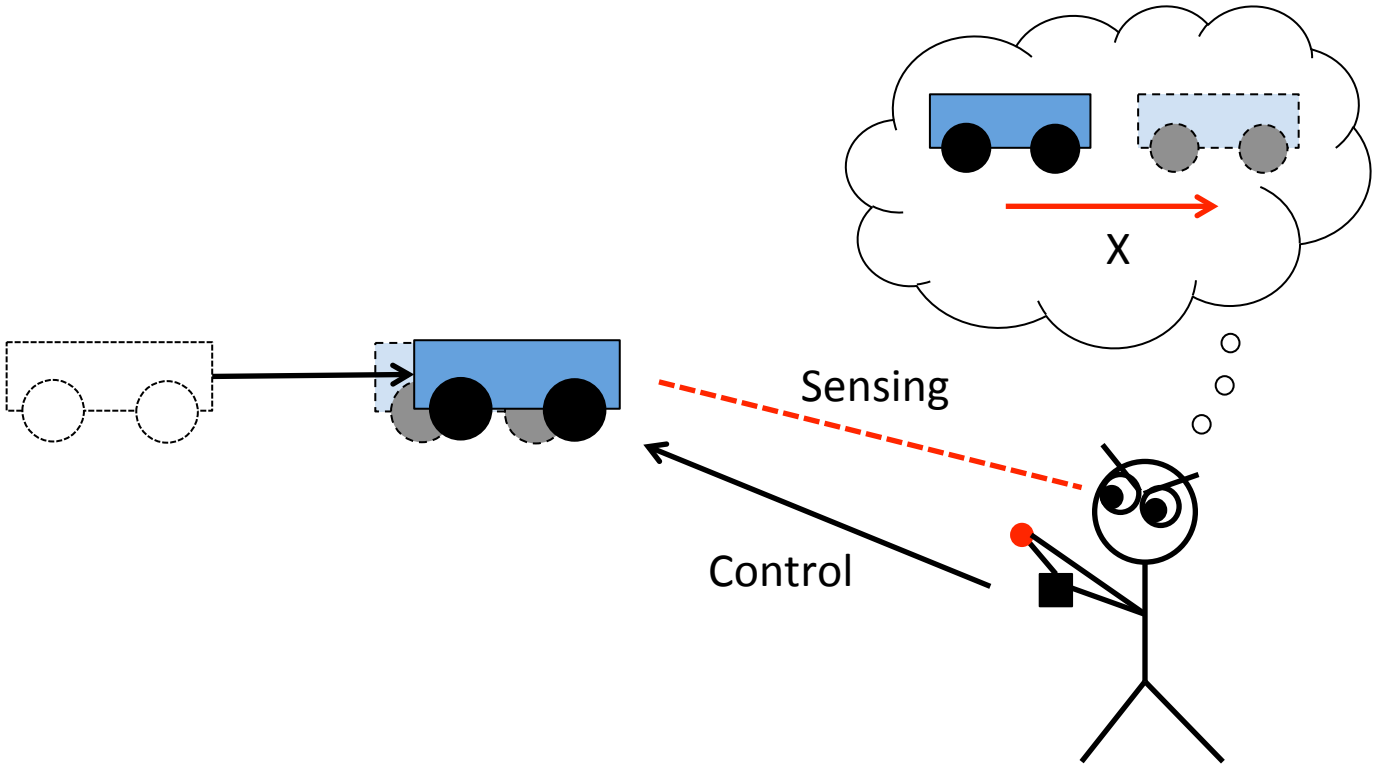
Closed Loop



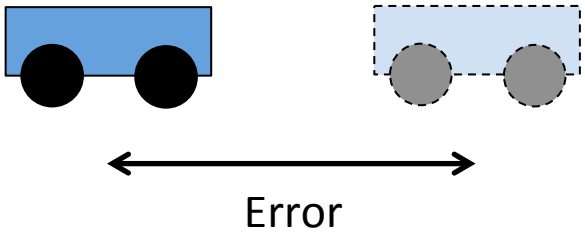
Closed Loop



Closed Loop

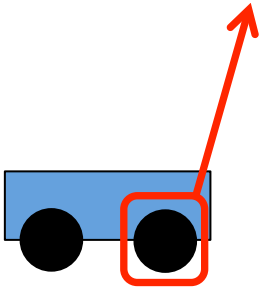


Some terminology

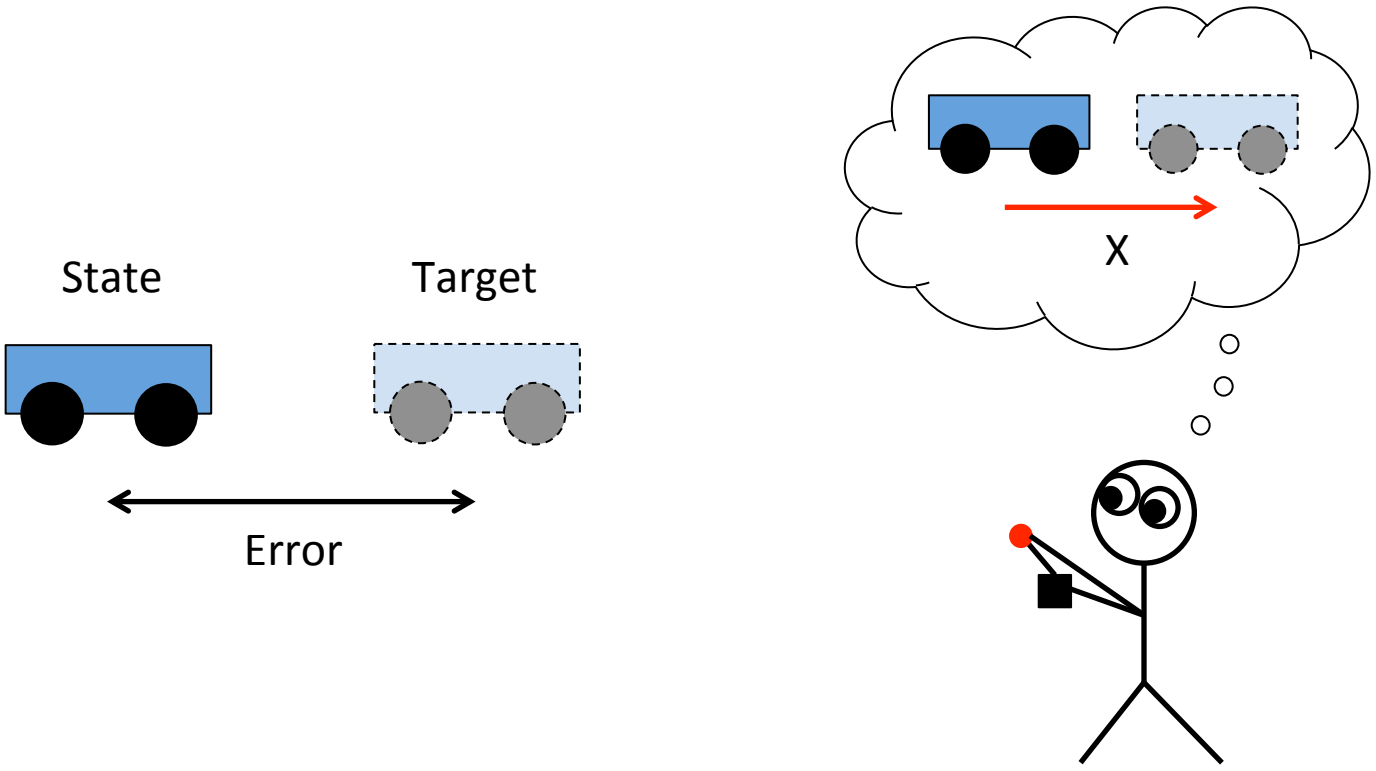


Motor

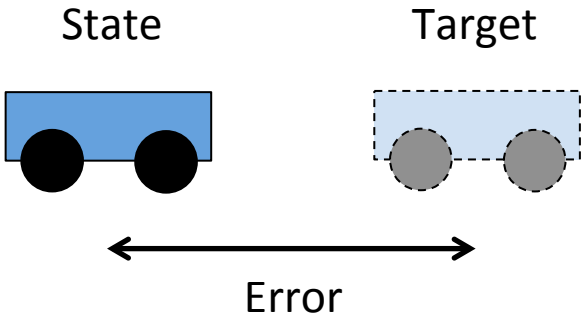
Encoder

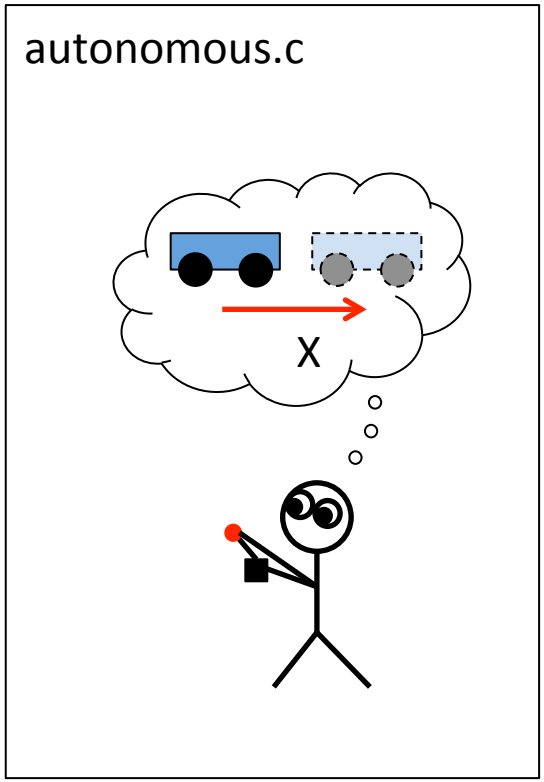


Autonomous Mode

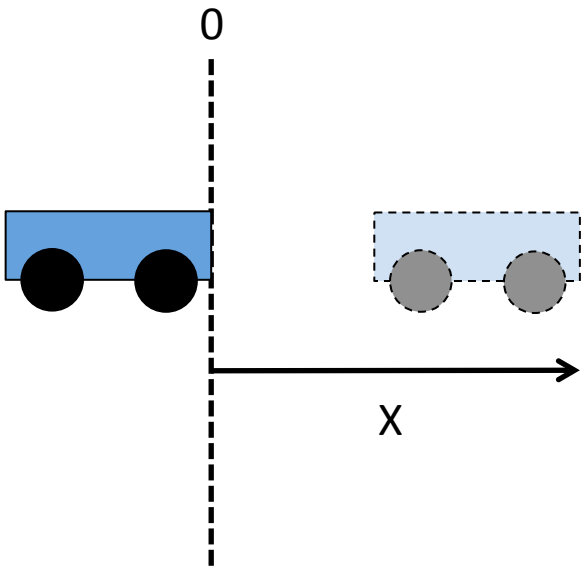


Autonomous Mode





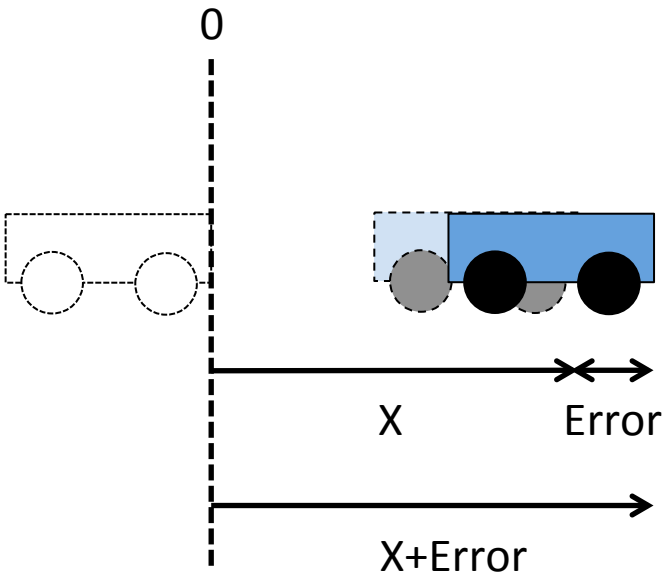
Autonomous Mode



autonomous.c

```
void go()
{
        ..
        robot.drive(X);
        ..
}
```

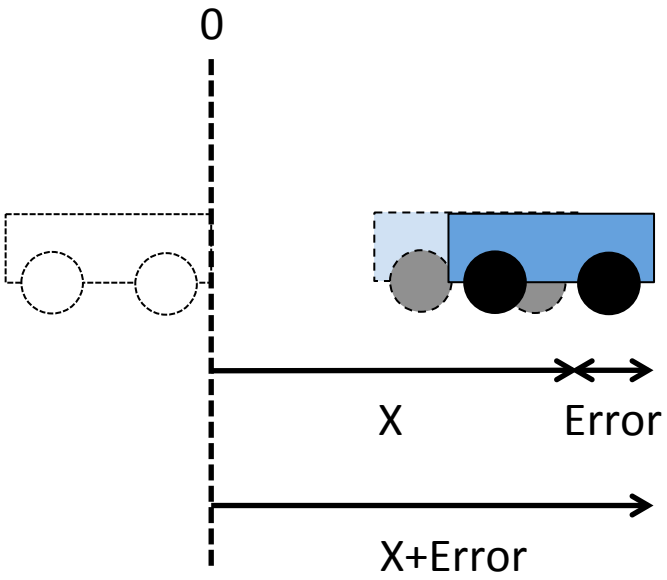
Autonomous Mode



autonomous.c

```
void go()
{
        ..
        robot.motors(100);
        wait(1_SEC);
        robot.motors(0);
        ..
}
```

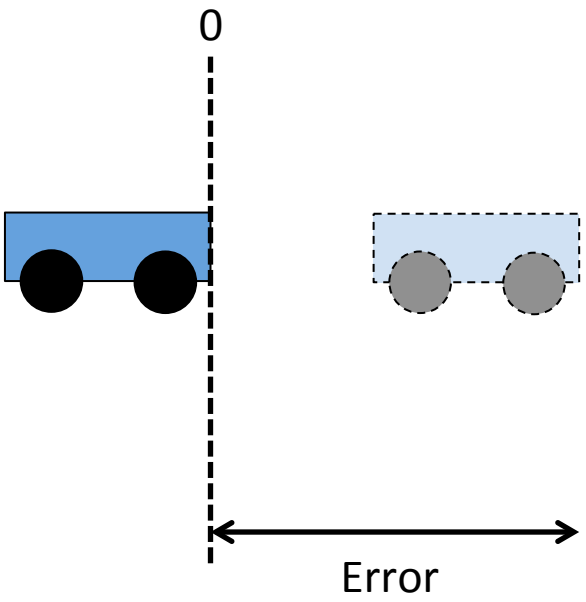
Autonomous Mode



autonomous.c

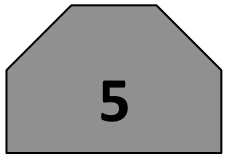
```
void go()
{
        ..
        robot.motors(100);

        while(encoder < 1000)
        {
          encoder = robot.read();
        }

        robot.motors(0);
        ..
}
```

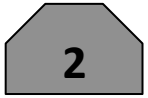
PID Control

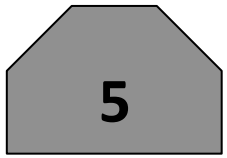


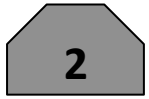
PID Control

Proportional		**I**ntegral		**D**erivative
Present		Past		Future
P*Error	**+**	**I***Sum_of_Errors	**+**	**D***(Error- Last_Error)



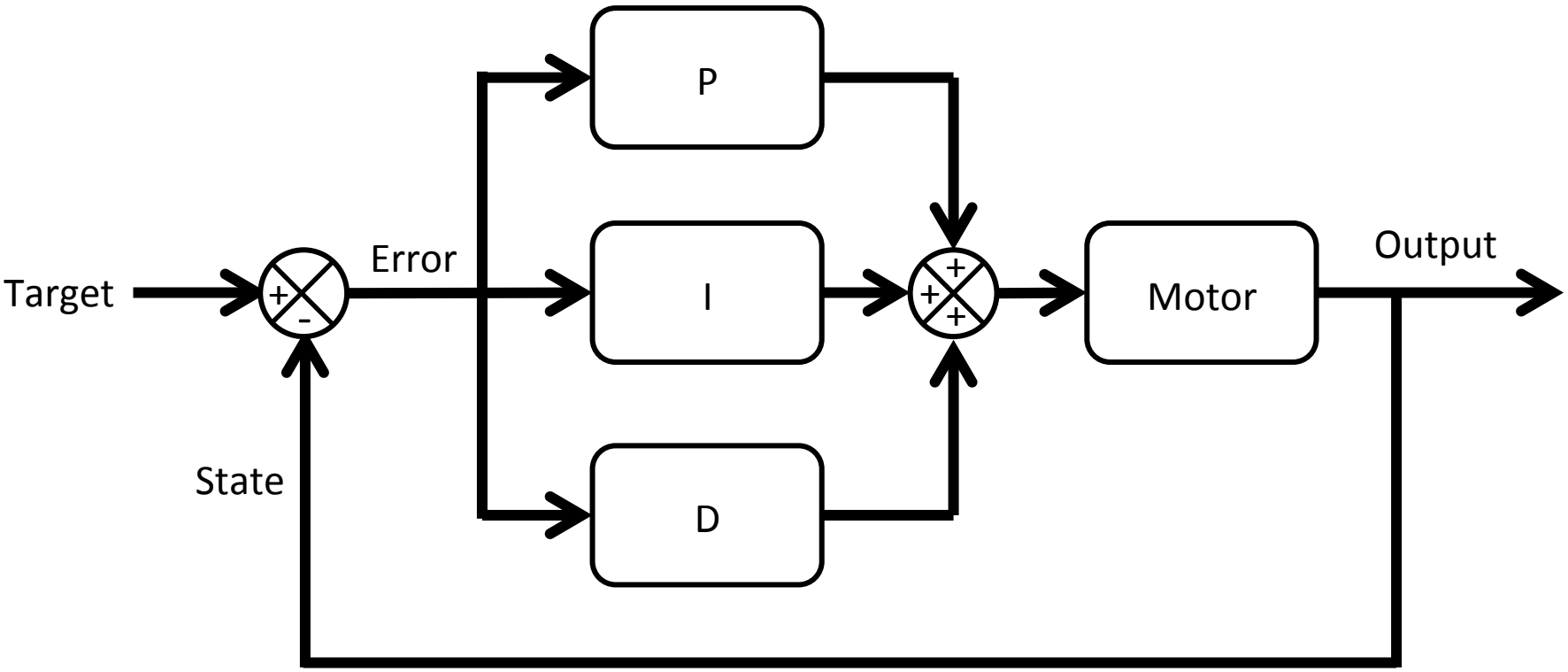




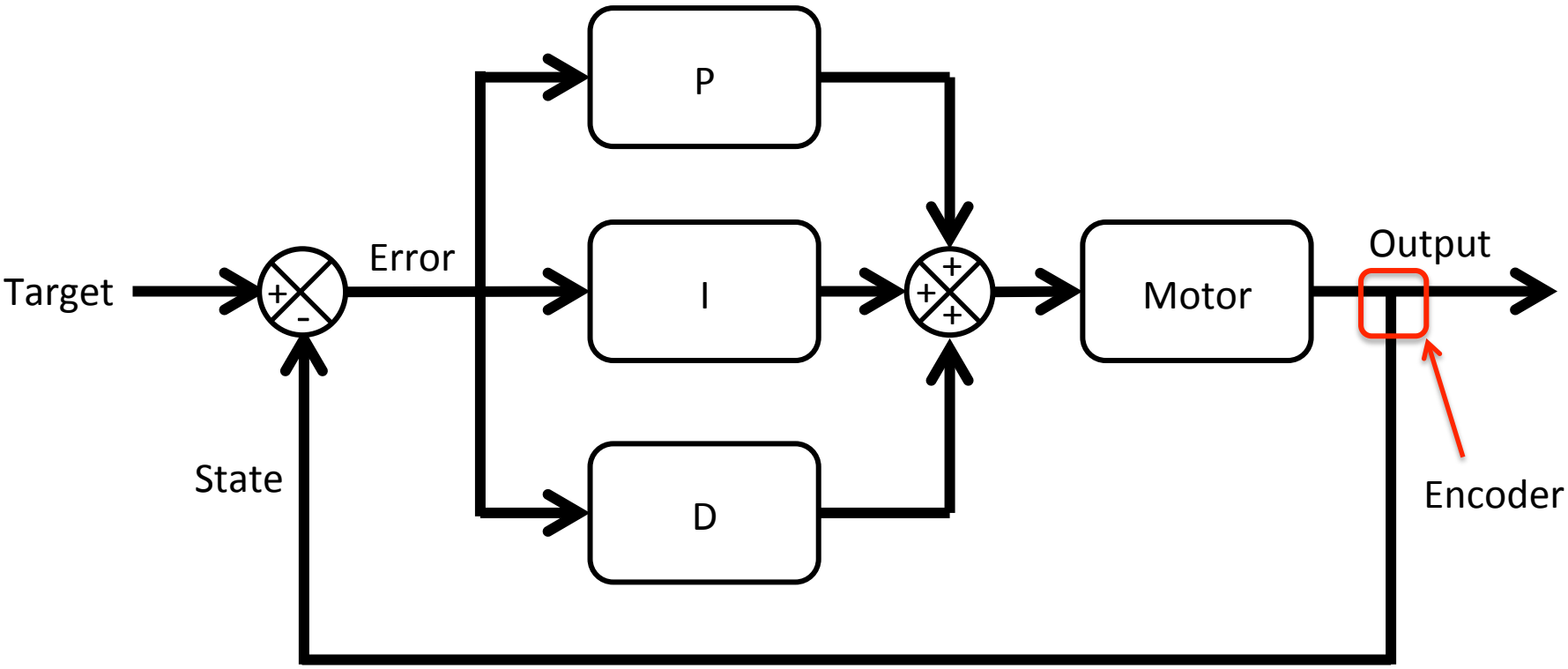


Position Control Using PID

PID Block Diagram

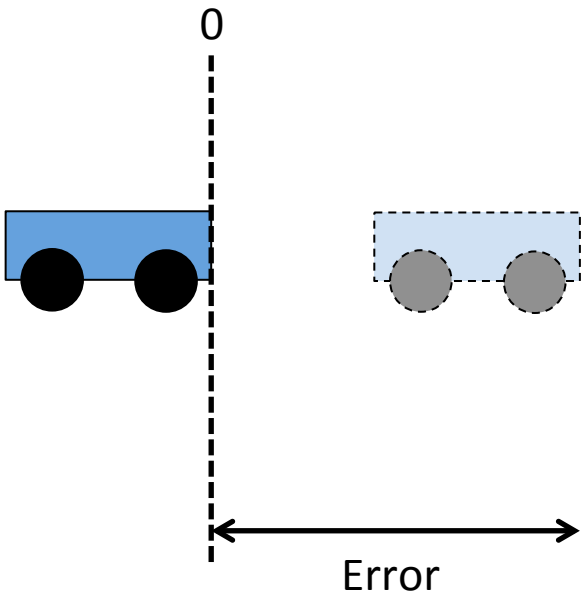


PID Block Diagram



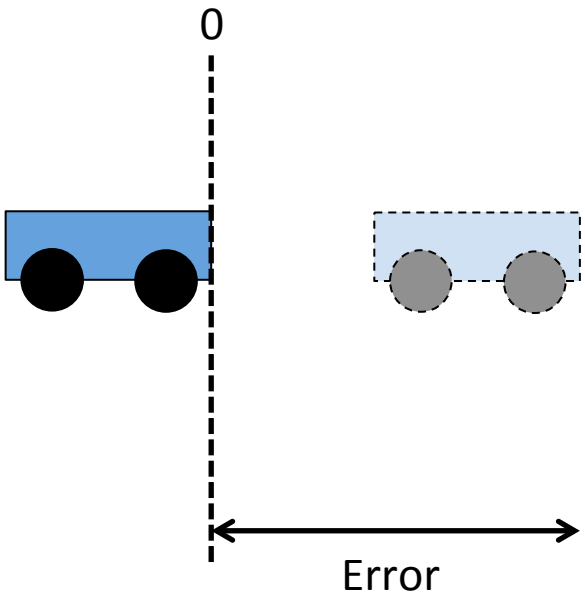
Proportional Control

Proportional Control



Motor = P * Error

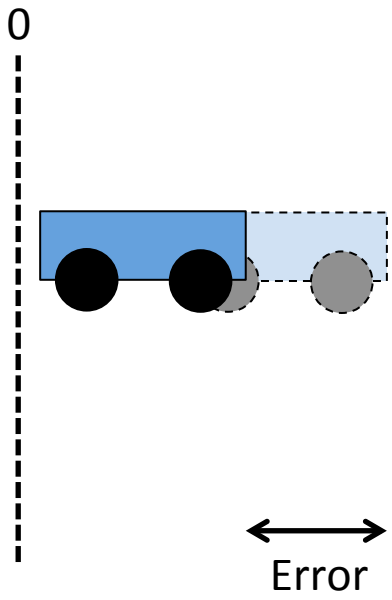
Proportional Control



Motor = P * Error

100 = 2 * 50

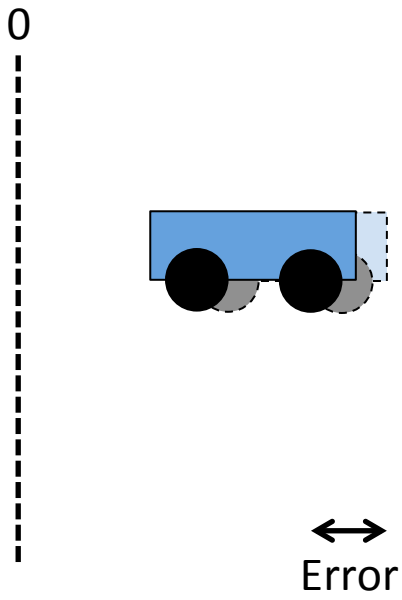
Proportional Control



Motor = P * Error

20 = 2 * 10

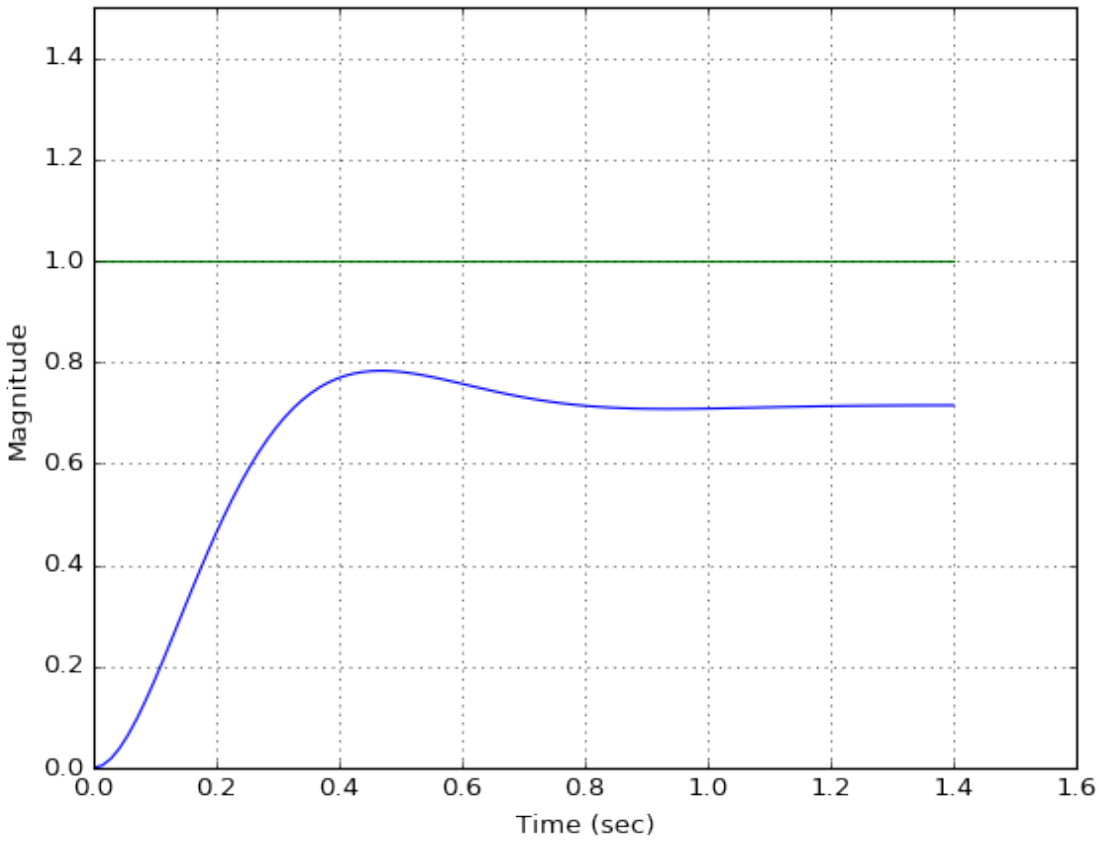
Proportional Control



Motor = P * Error

1 = 2 * 1

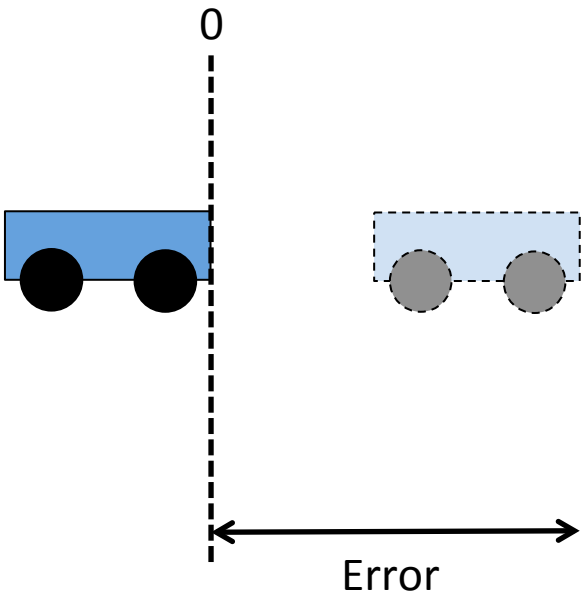
Proportional Control



Integral Control

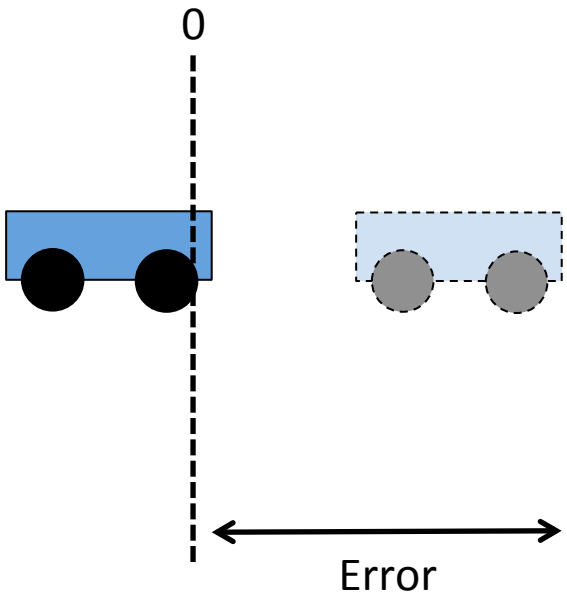
Integral Control

Error_Sum = Error_Sum + Error
50 = 0 + 50



Motor = I * Error_Sum
25 = 0.5 * 50

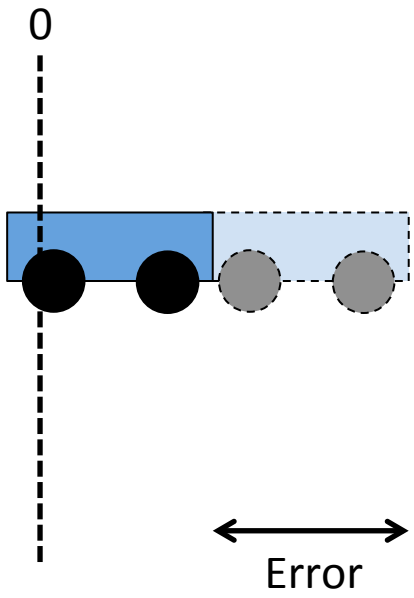
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95

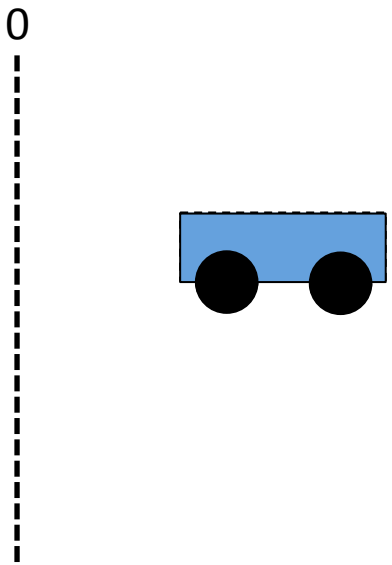
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115

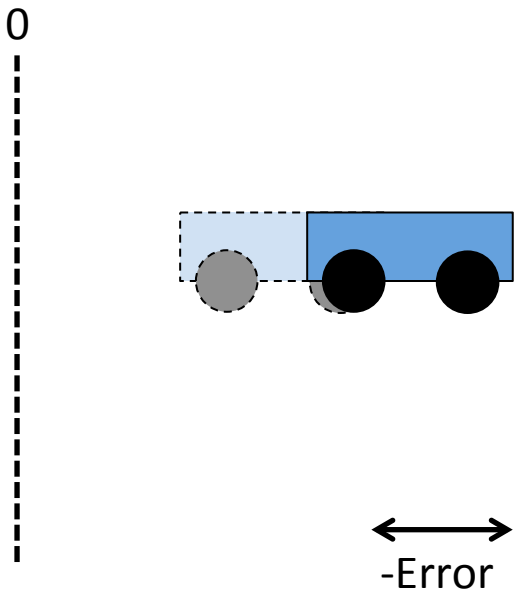
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115

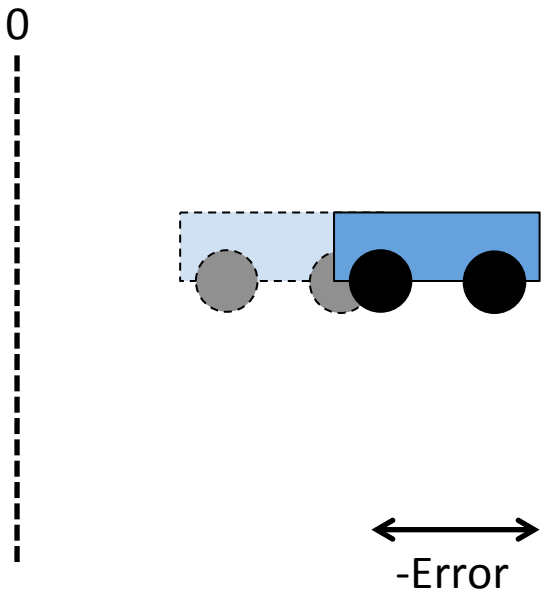
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0
75	= 115 + -40

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115
38	= 0.5 * 75

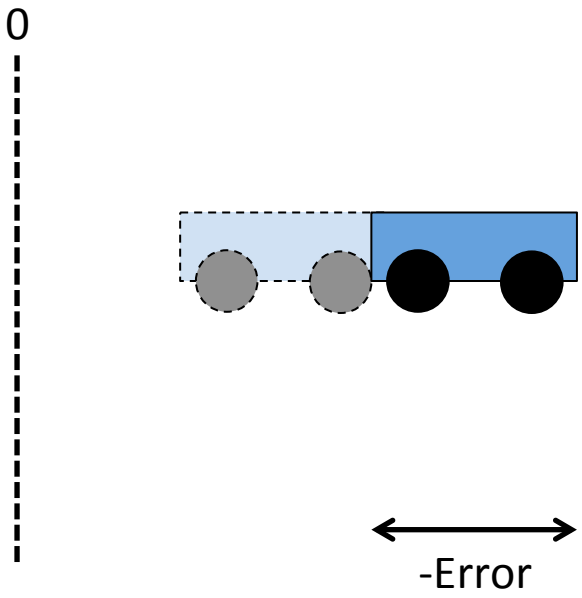
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0
75	= 115 + -40
25	= 75 + -50

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115
38	= 0.5 * 75
13	= 0.5 * 25

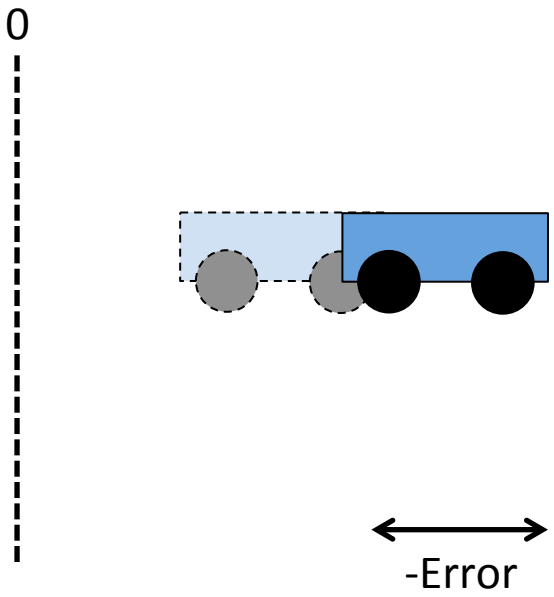
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0
75	= 115 + -40
25	= 75 + -50
-35	= 25 + -60

Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115
38	= 0.5 * 75
13	= 0.5 * 25
-17	= 0.5 * -35

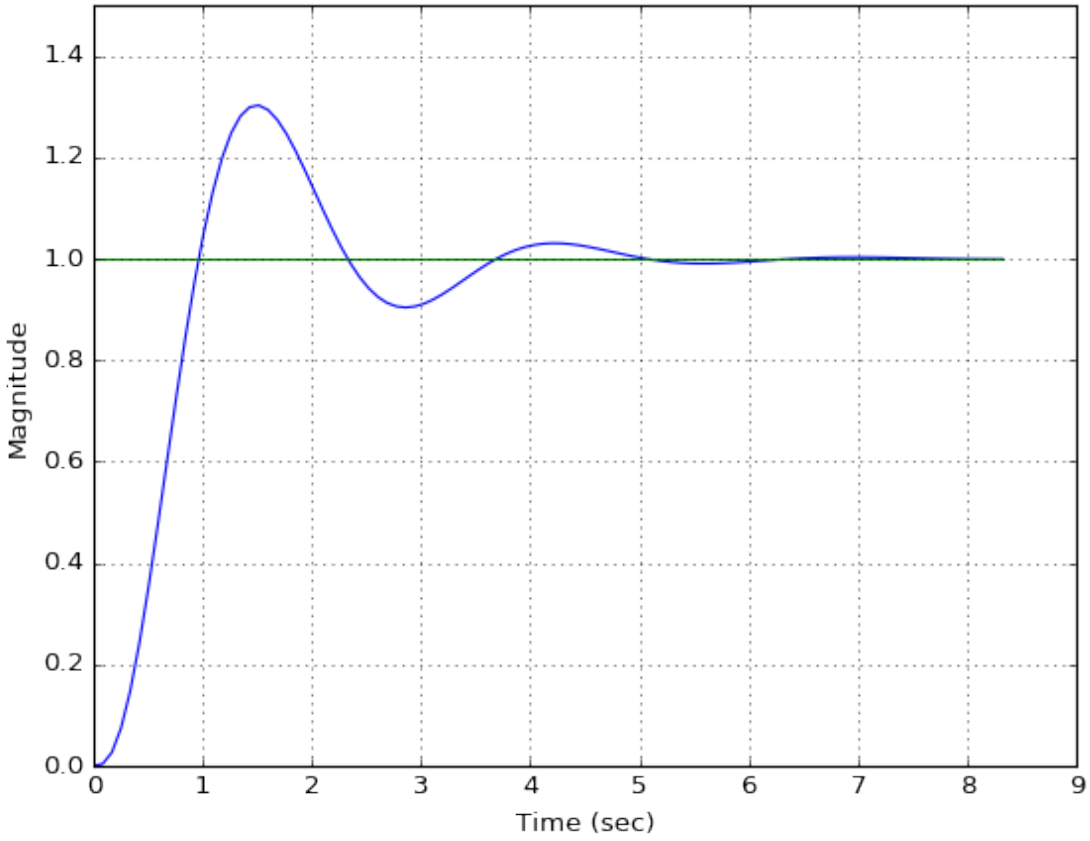
Integral Control



Error_Sum	**= Error_Sum + Error**
50	= 0 + 50
95	= 50 + 45
115	= 95 + 20
115	= 115 + 0
75	= 115 + -40
25	= 75 + -50
-35	= 25 + -60
-75	= -35 + -40

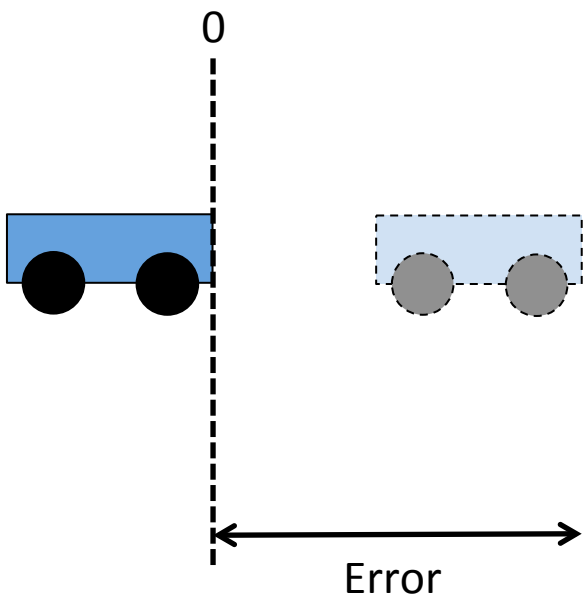
Motor	**= I * Error_Sum**
25	= 0.5 * 50
48	= 0.5 * 95
56	= 0.5 * 115
56	= 0.5 * 115
38	= 0.5 * 75
13	= 0.5 * 25
-17	= 0.5 * -35
-38	= 0.5 * -75

Integral Control



Derivative Control

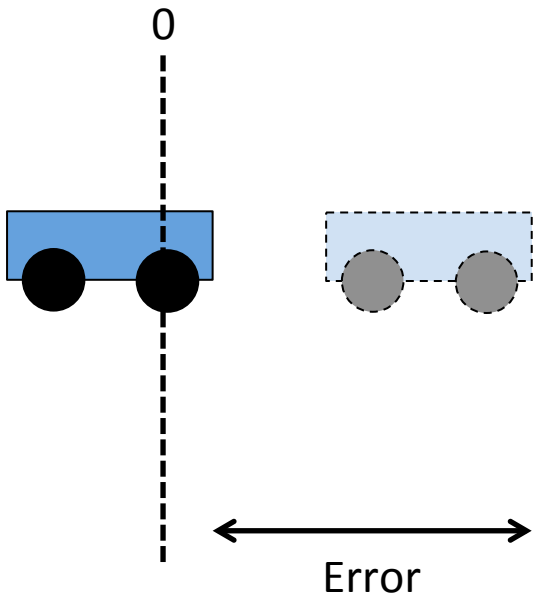
Derivative Control



Motor = D* (Error – Last_Error)

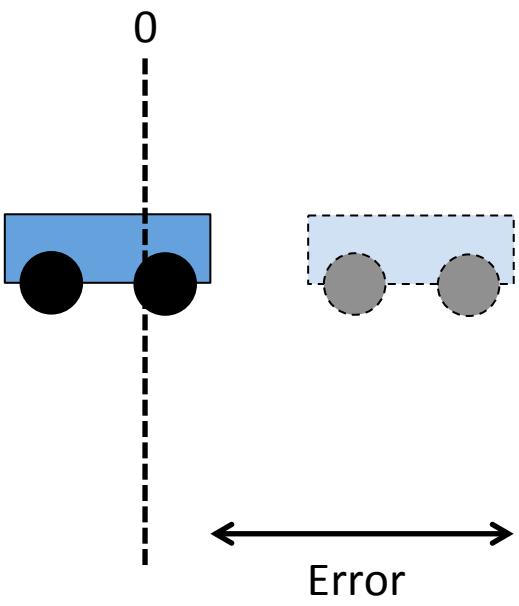
25 = 0.5 * (50 – 0)

Derivative Control



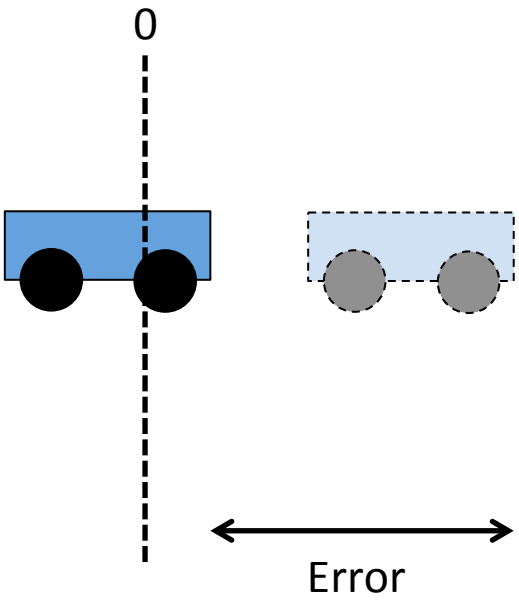
Motor	**= D* (Error – Last_Error)**
25	= 0.5 * (50 – 0)
-5	= 0.5 * (40 – 50)

Derivative Control



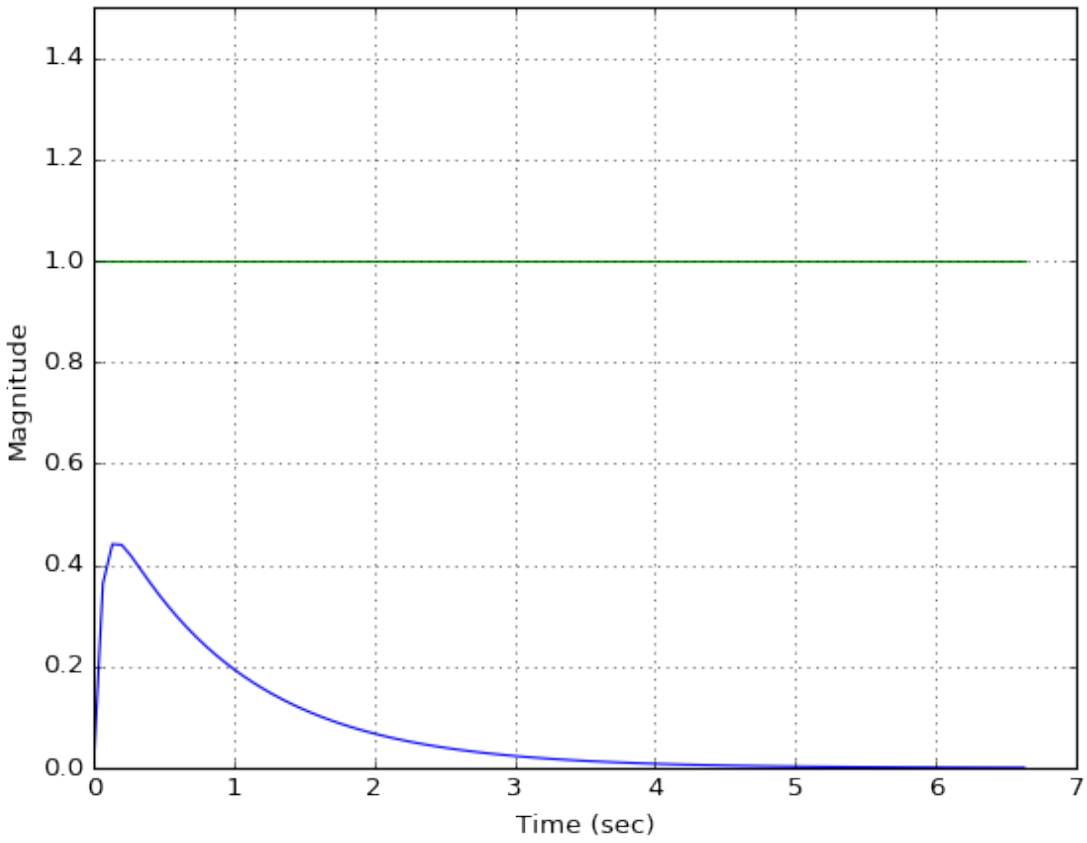
Motor	**= D* (Error – Last_Error)**
25	= 0.5 * (50 – 0)
-5	= 0.5 * (40 – 50)
-2.5	= 0.5 * (35 – 40)

Derivative Control



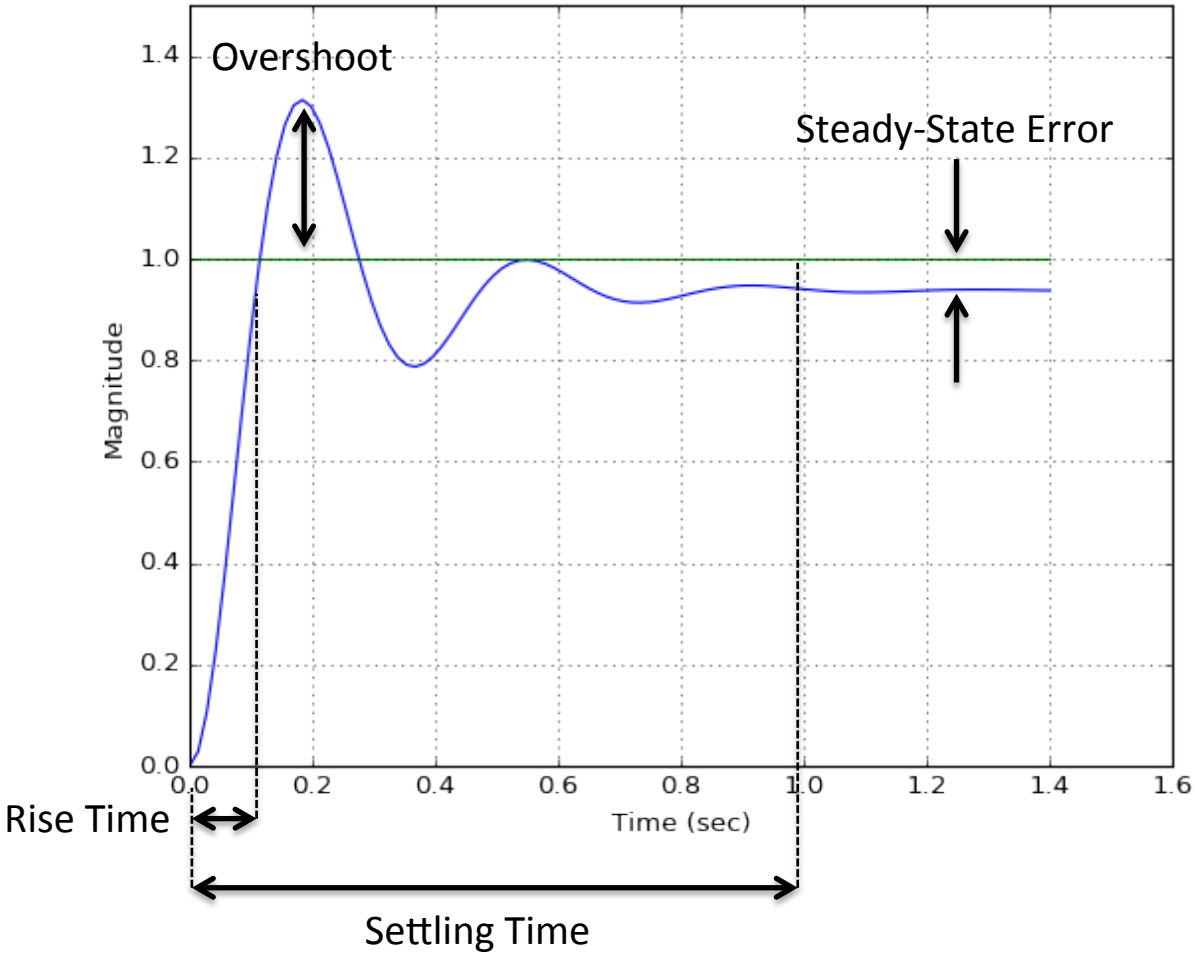
Motor	**= D* (Error – Last_Error)**
25	= 0.5 * (50 – 0)
-5	= 0.5 * (40 – 50)
-2.5	= 0.5 * (35 – 40)
0	= 0.5 * (35 – 35)

Derivative Control

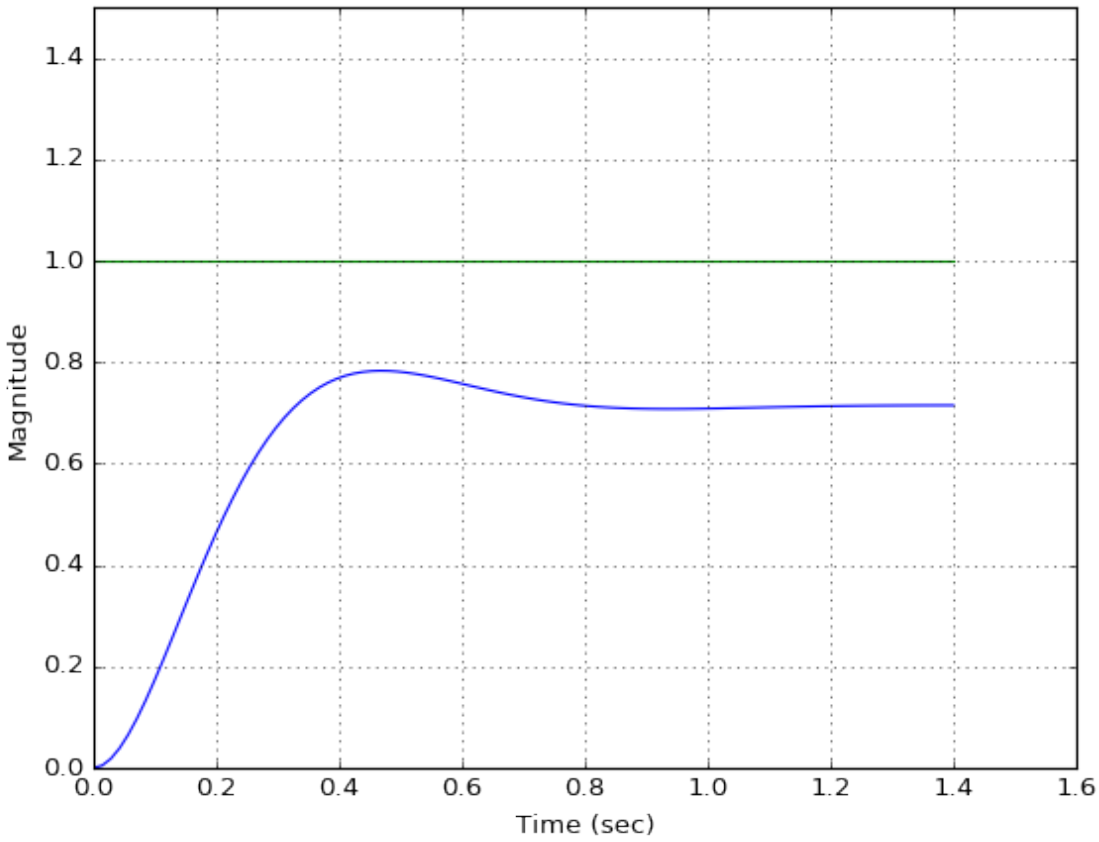


PID Plot Terminology

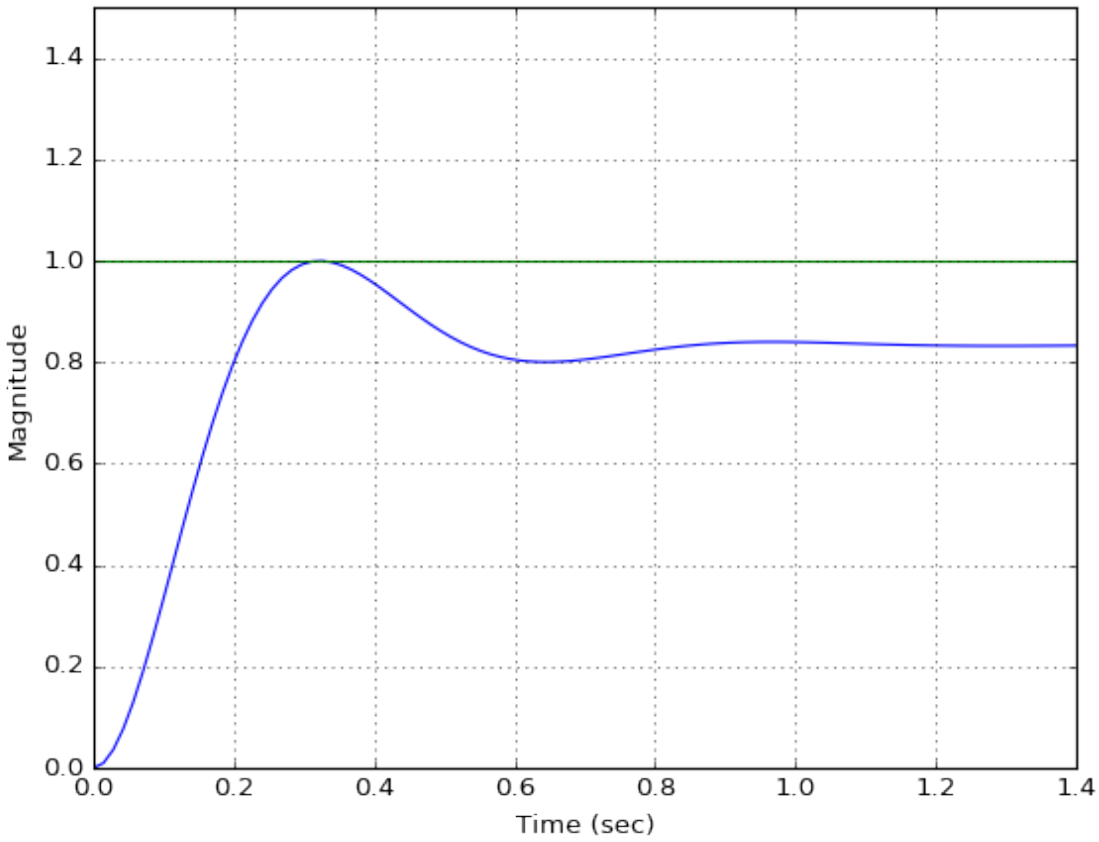
PID Plot Terminology



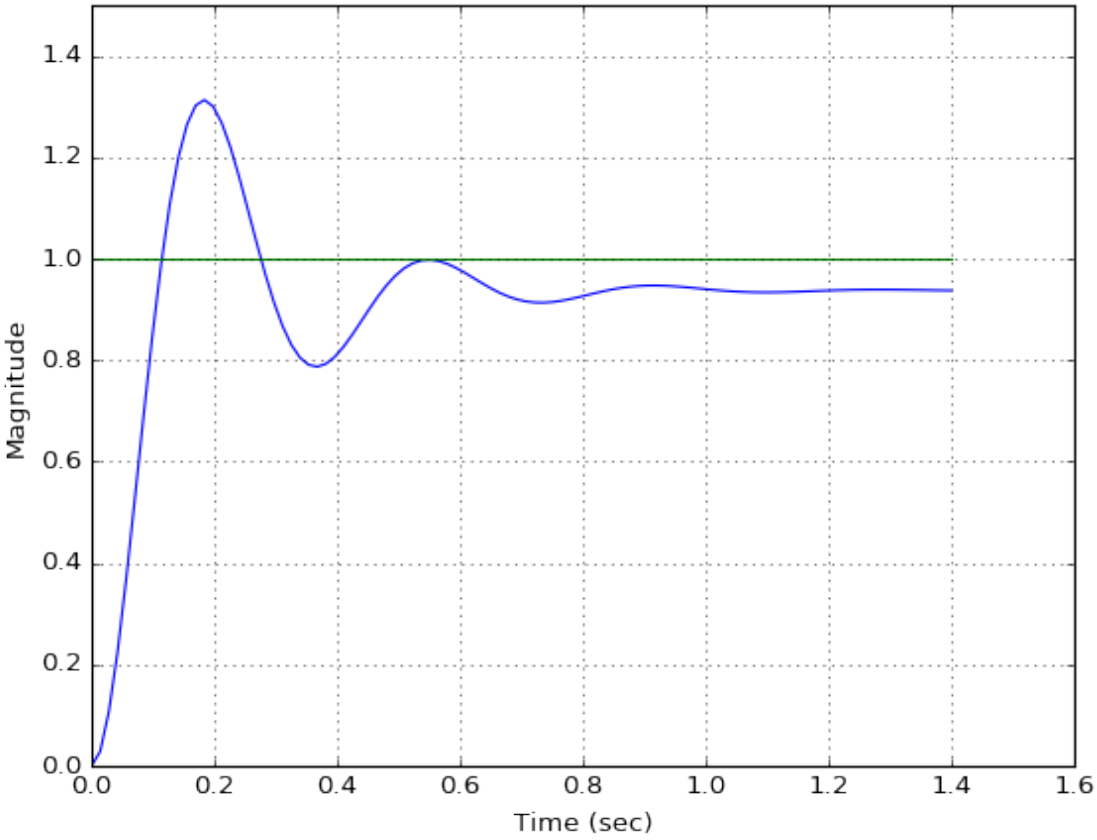
P = 50


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Magnitude
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (sec)

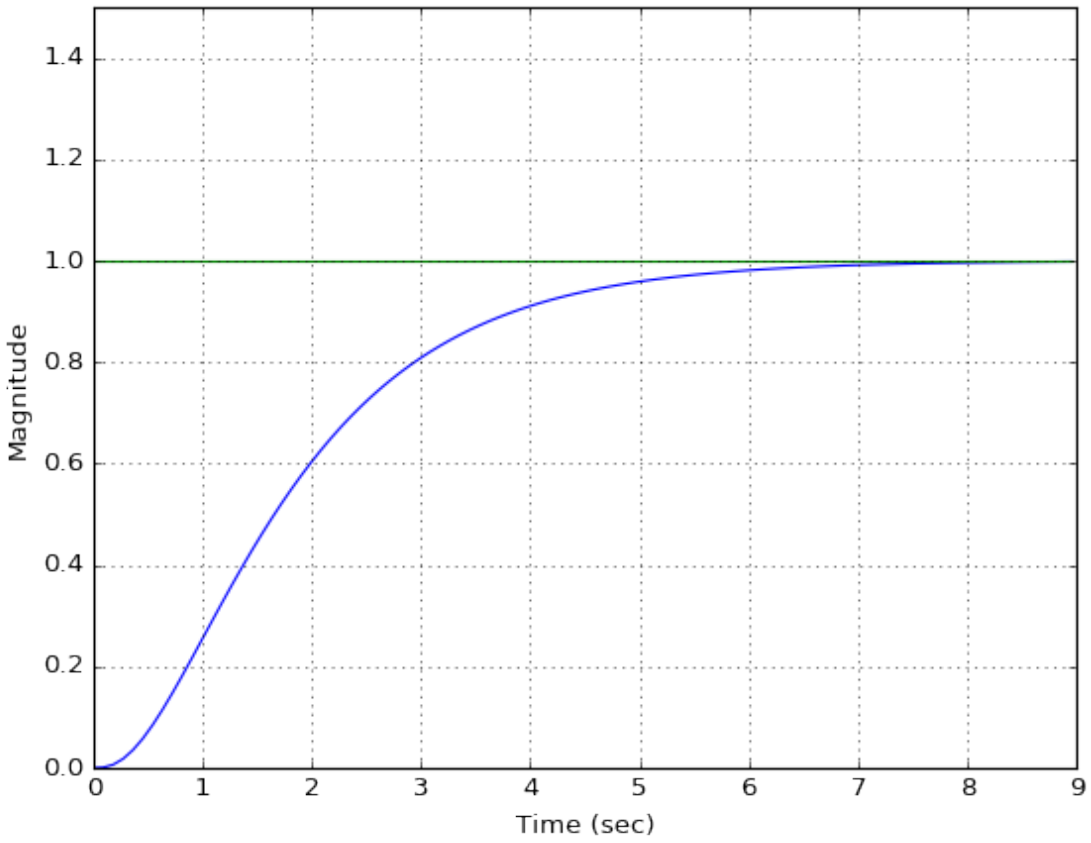
P = 100



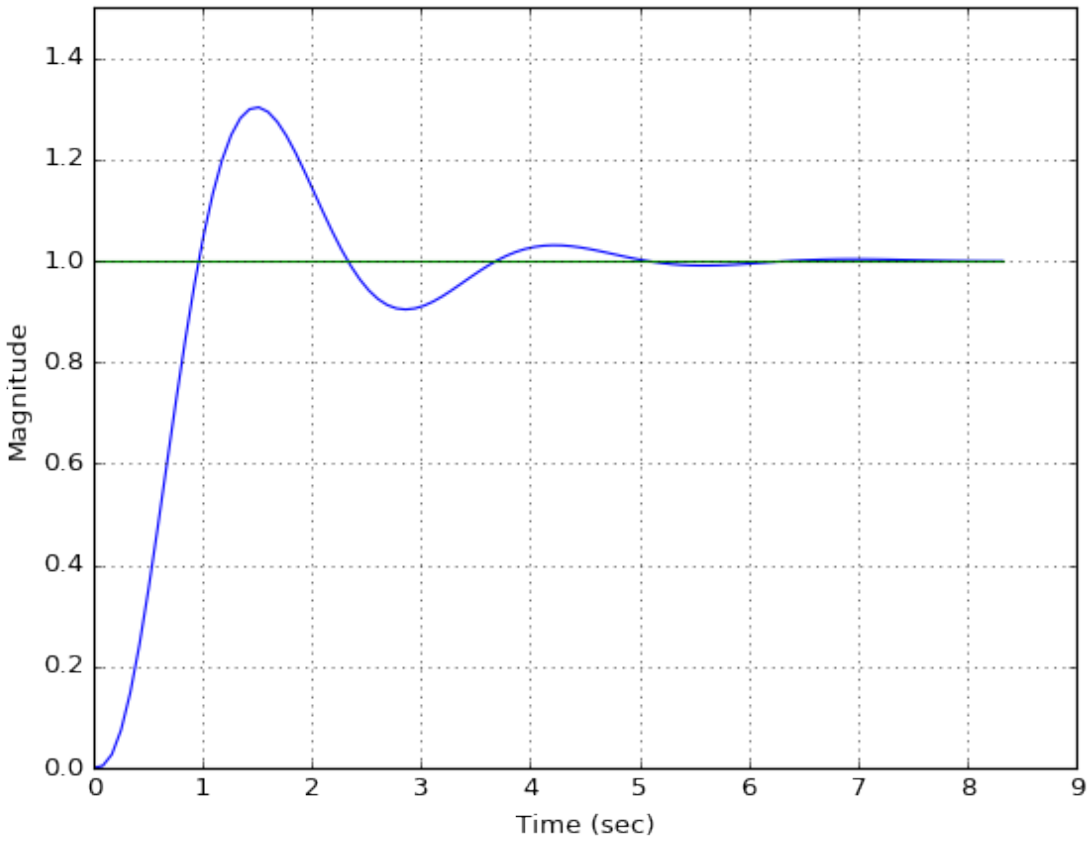
P = 300


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Magnitude
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (sec)

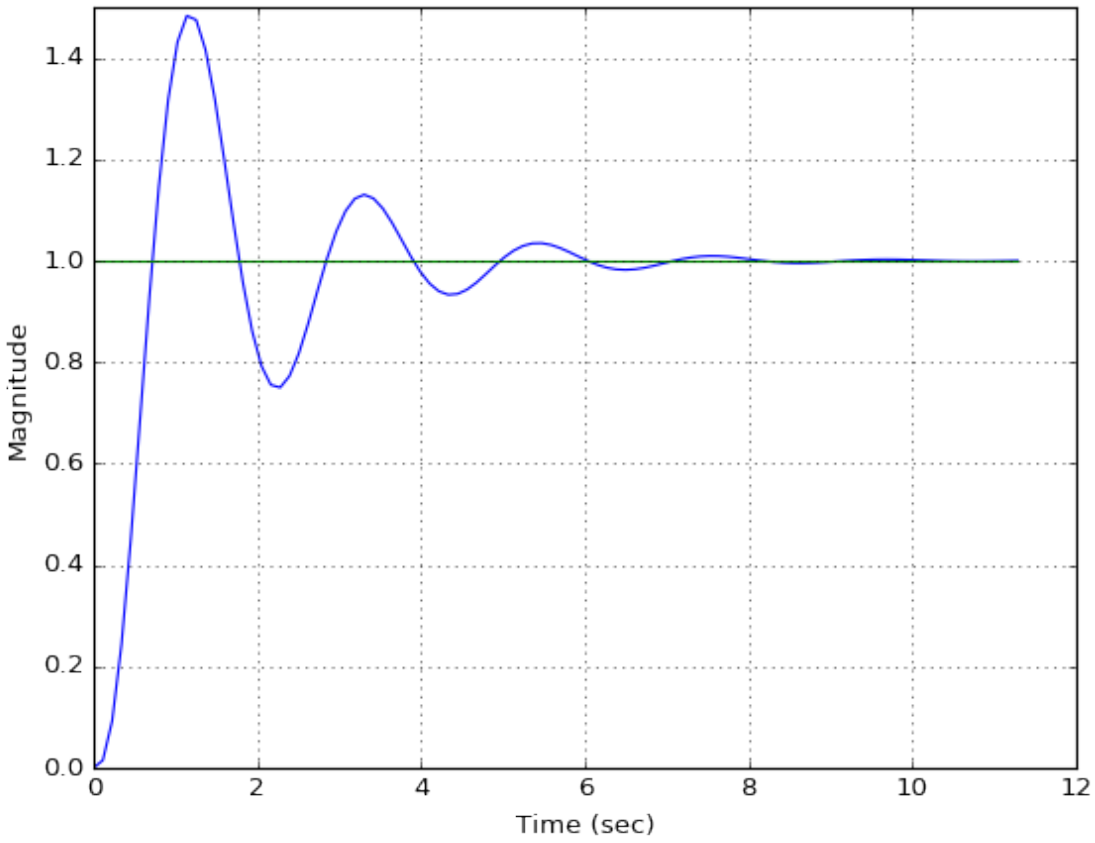
I = 10



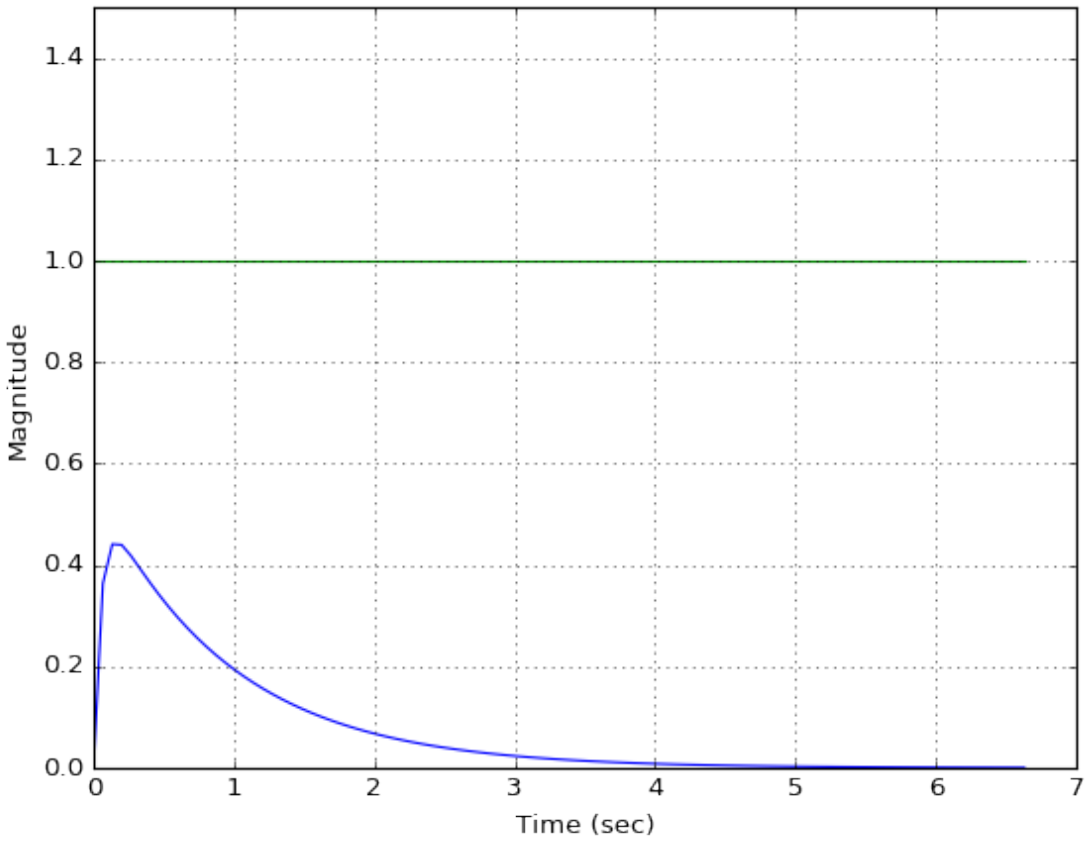
I = 50



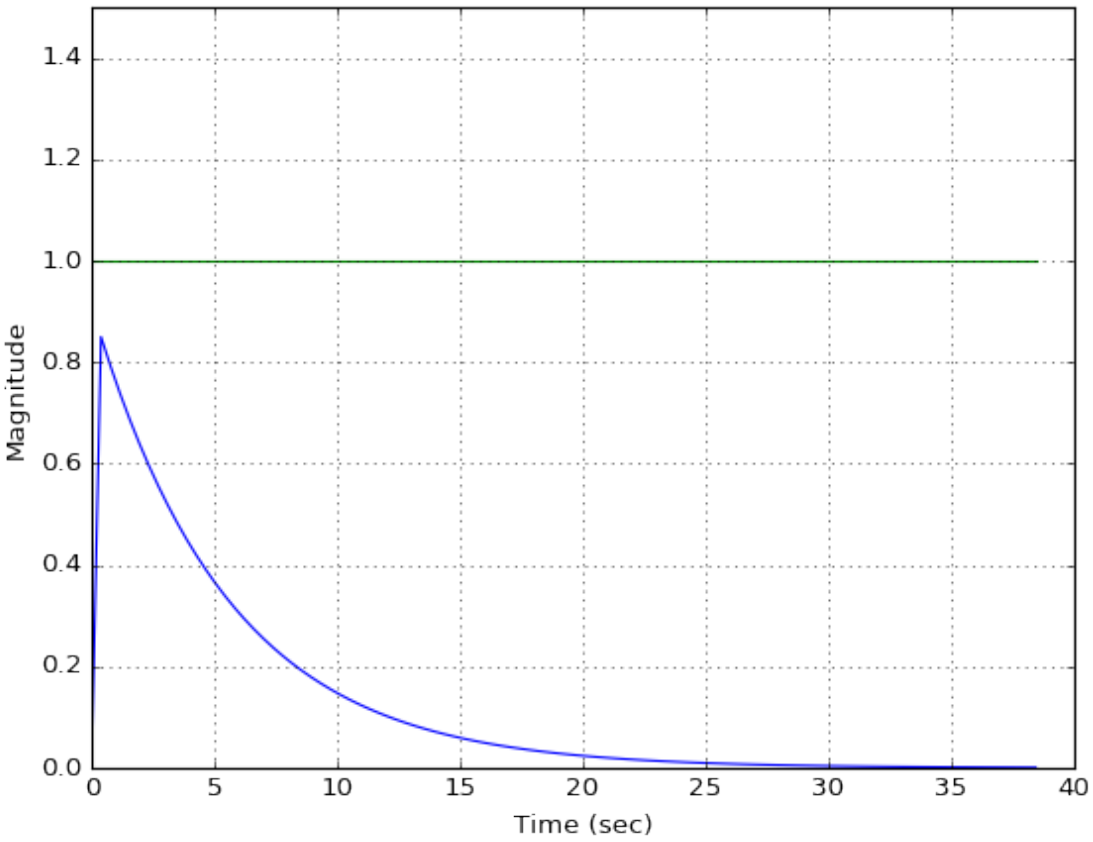
I = 80



D = 10

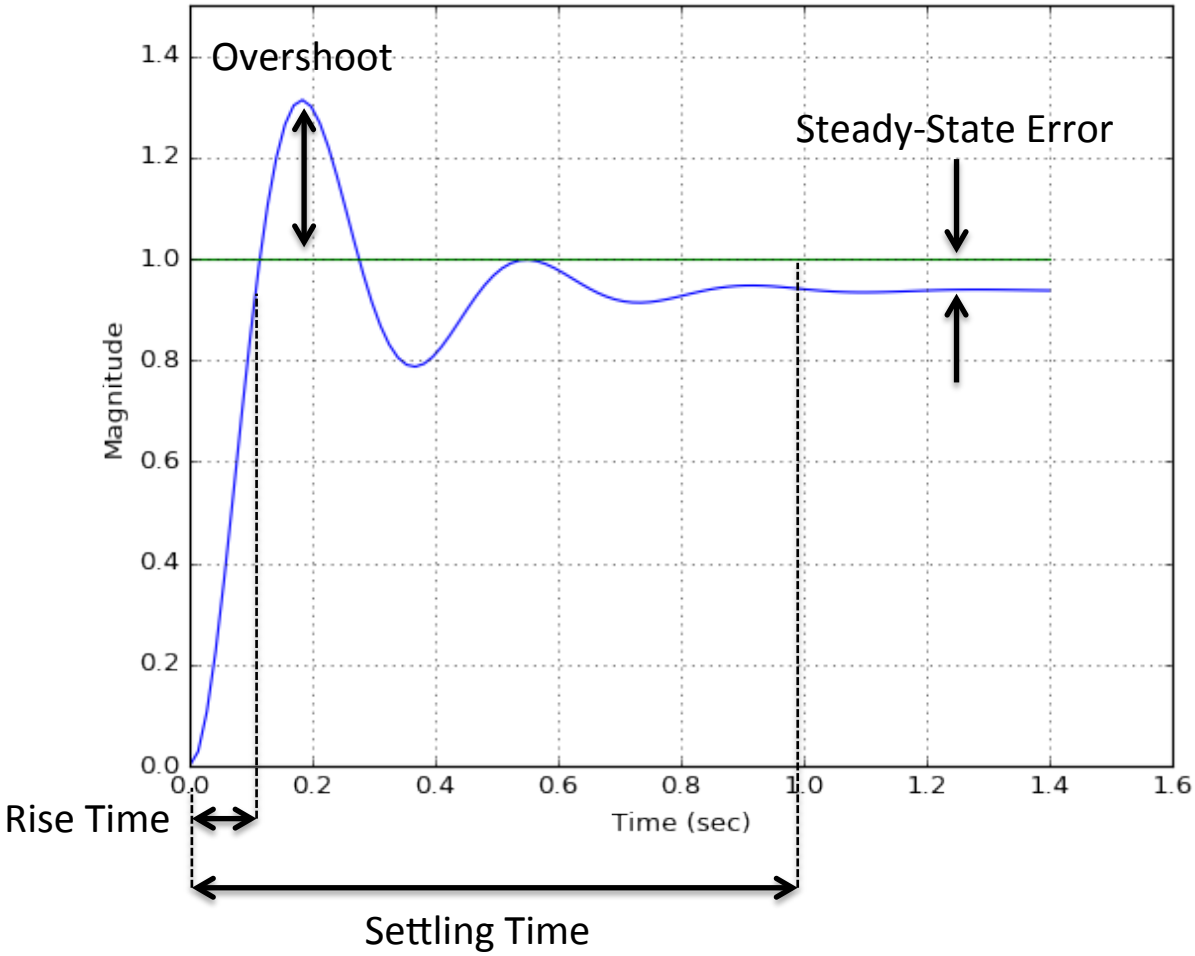


D = 100



PID Tuning Methods

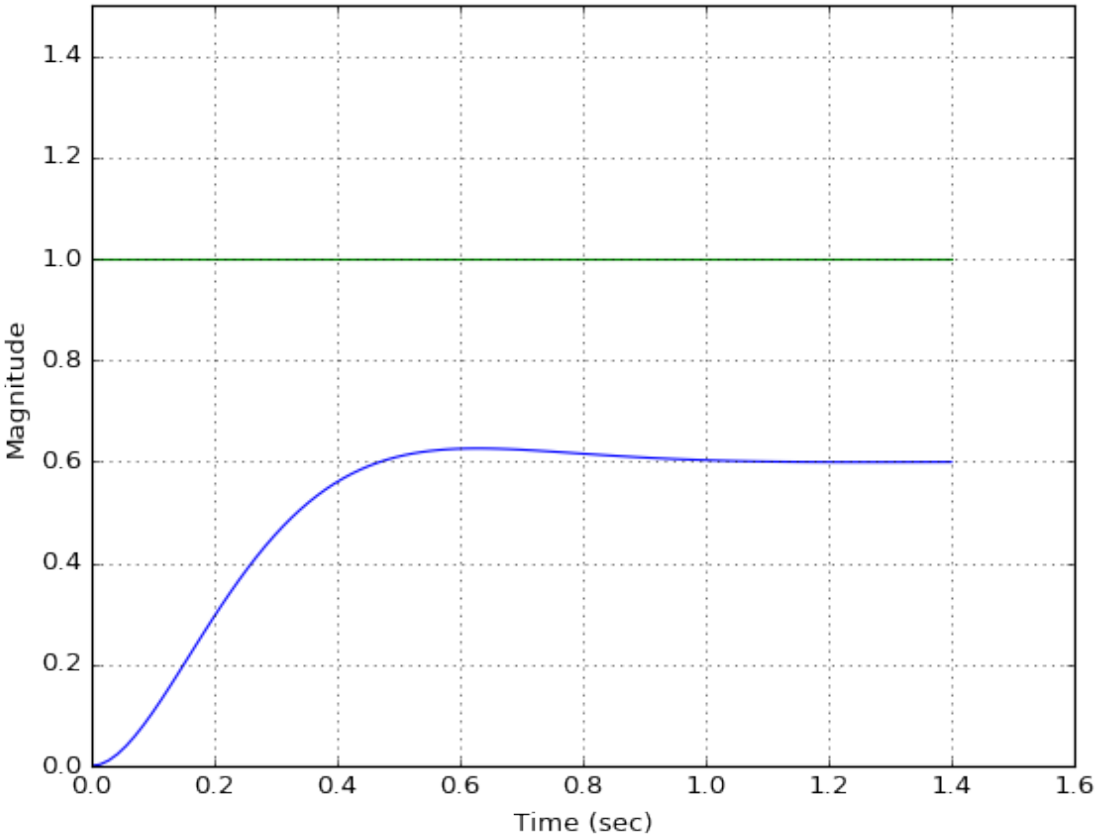
PID Plot Terminology



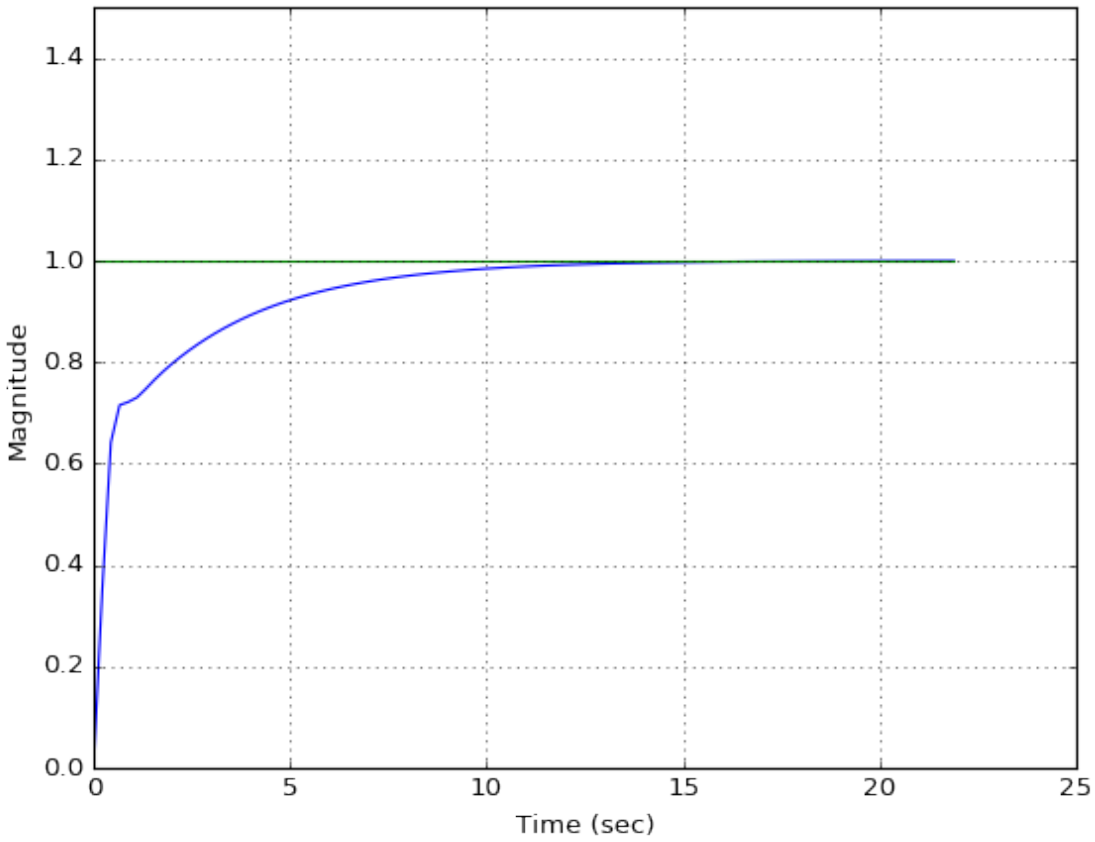
PID Tuning Methods

Parameter Increase	Rise Time	Overshoot	Settling Time	Steady-State Error
Kp	↓	↑	Small Change	↓
Ki	↓	↑	↑	↓ ↓
Kd	Small Change	↓	↓	Small Change

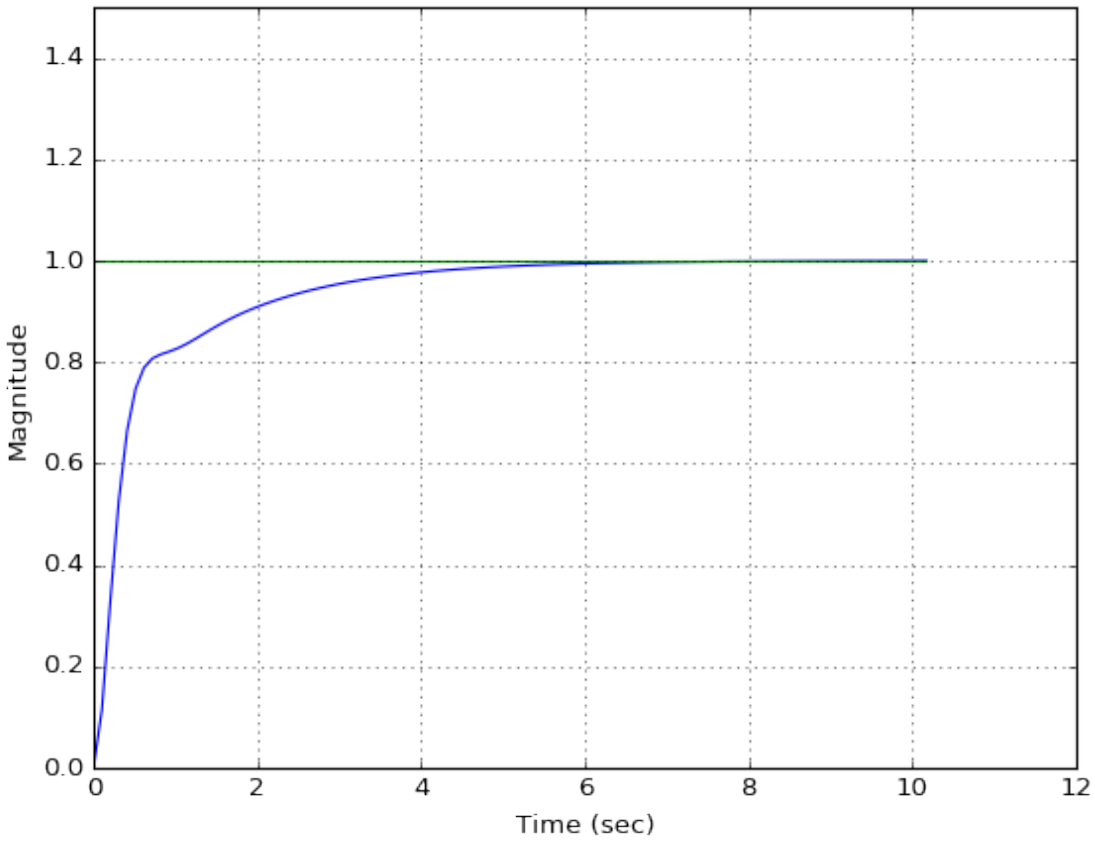
P = 30, I = 0, D = 0


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Magnitude
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (sec)

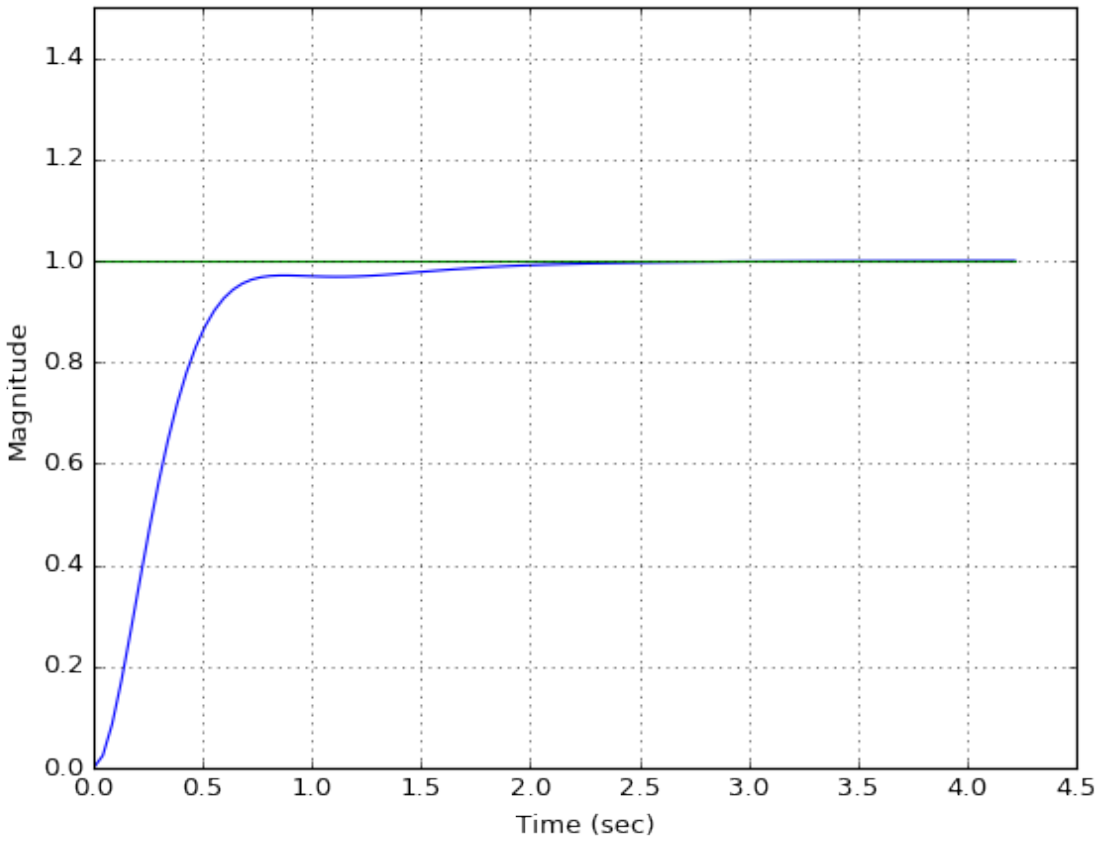
P = 30, I = 15, D = 0



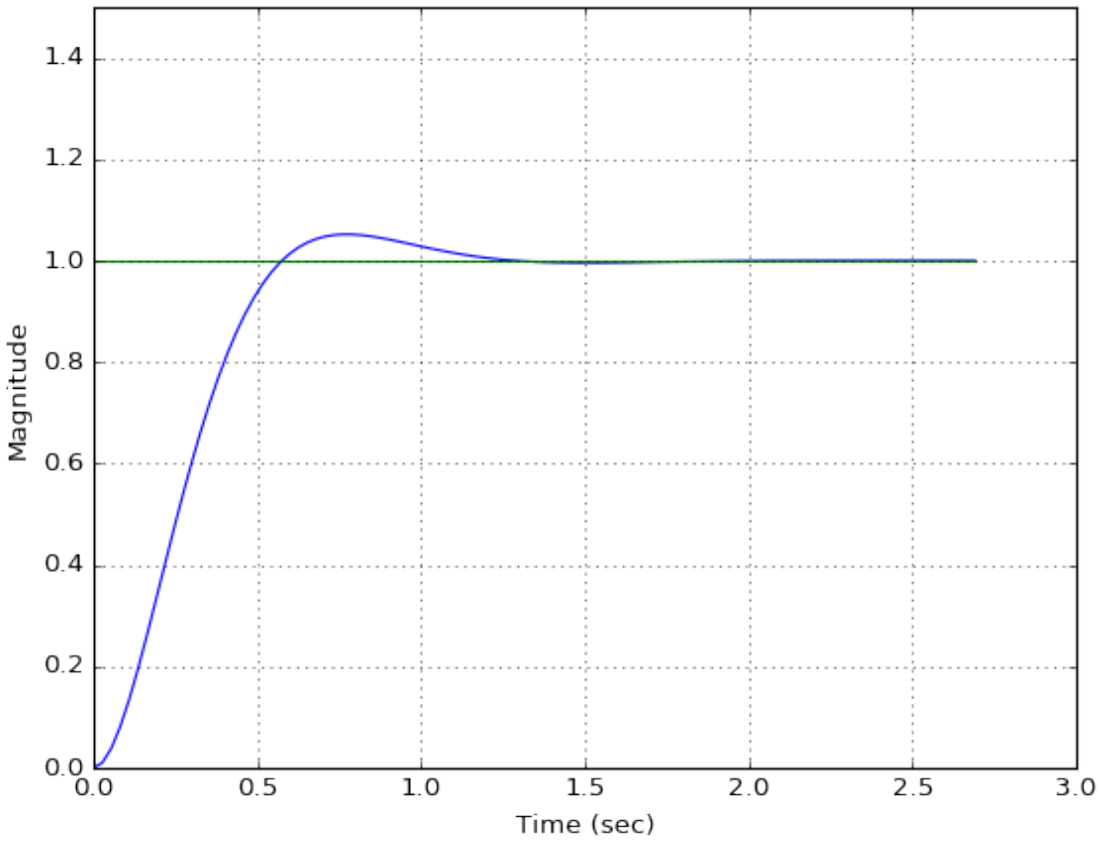
P = 30, I = 30, D = 0



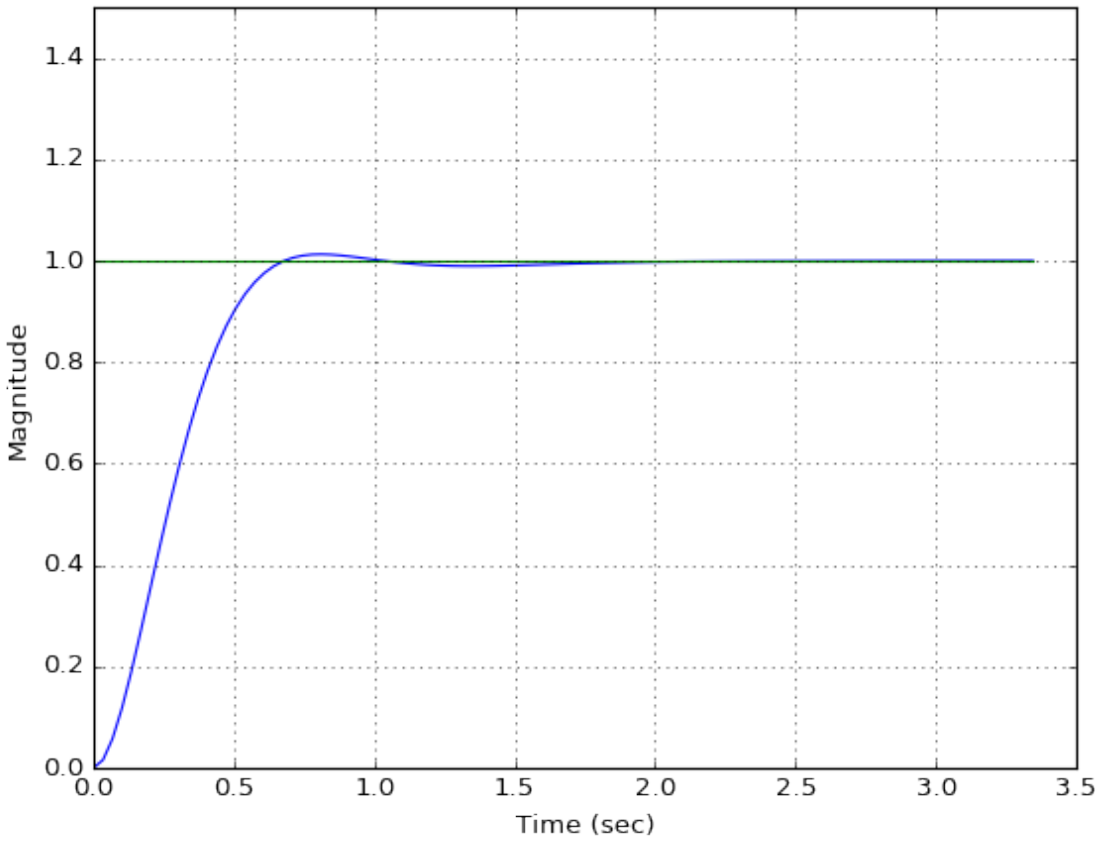
P = 30, I = 60, D = 0


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Magnitude
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Time (sec)

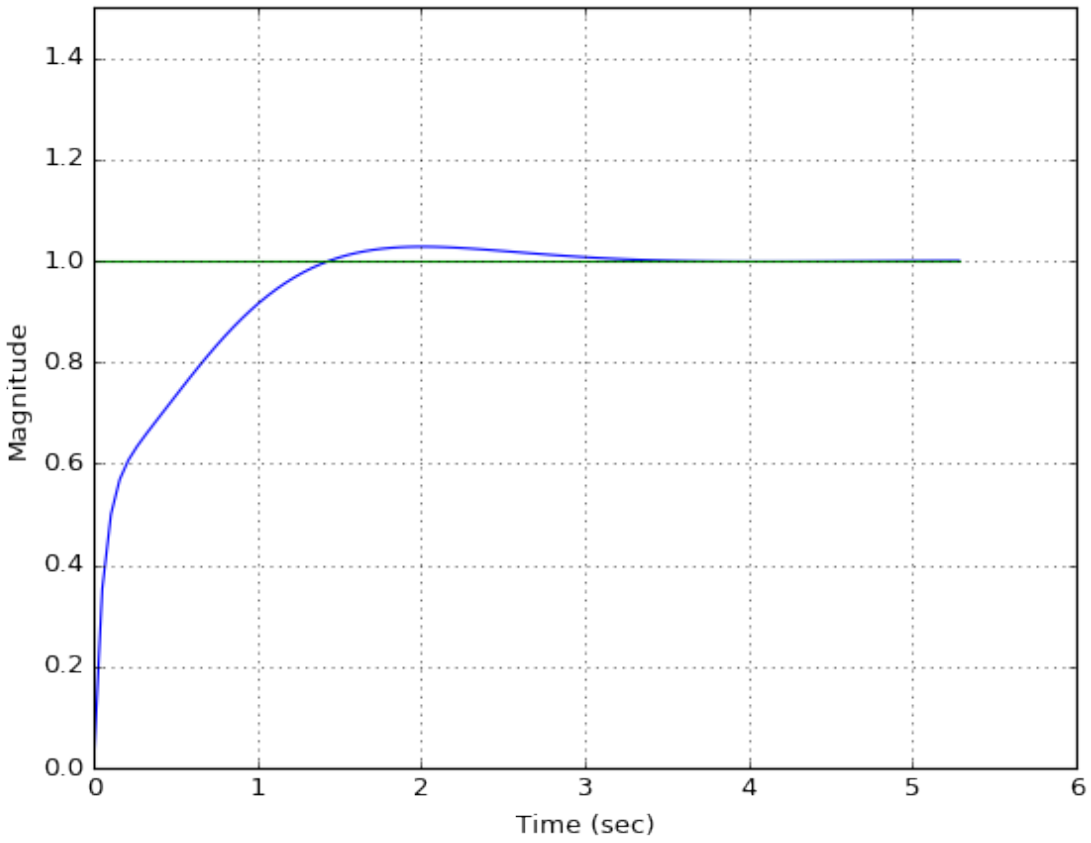
P = 30, I = 80, D = 0



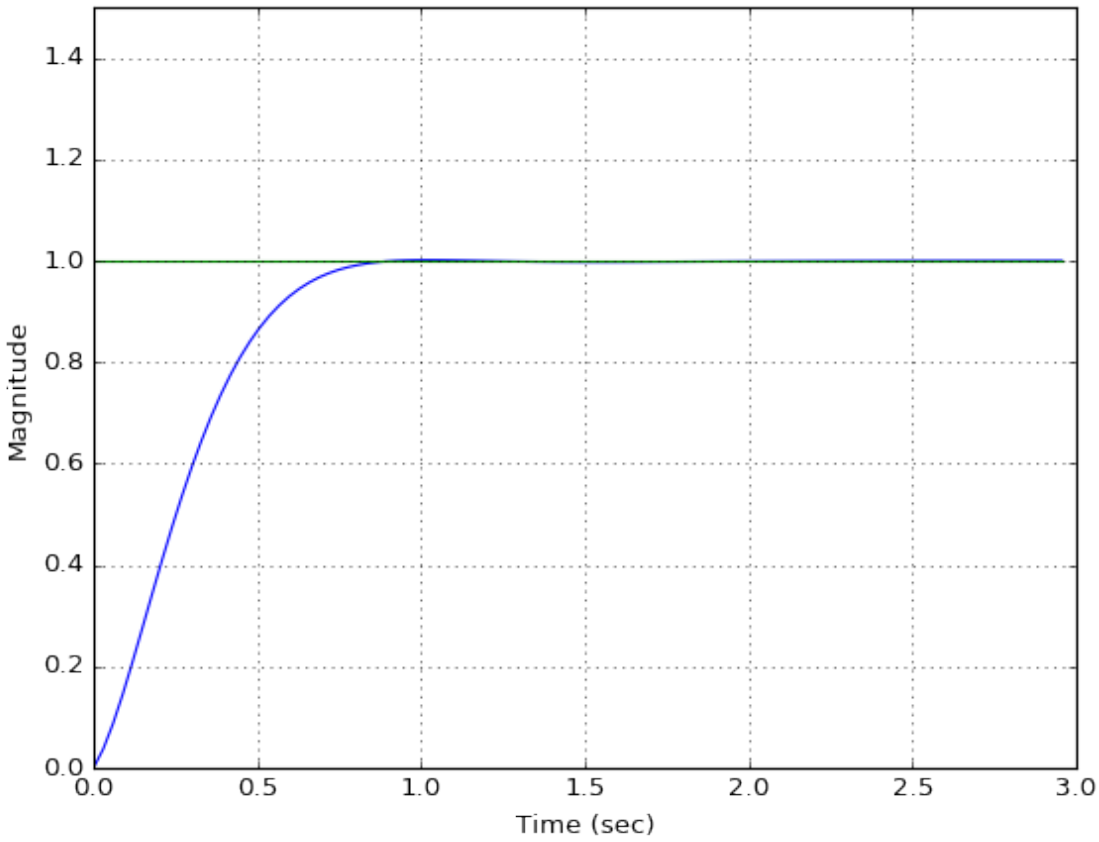
P = 30, I = 70, D = 0



P = 30, I = 70, D = 10

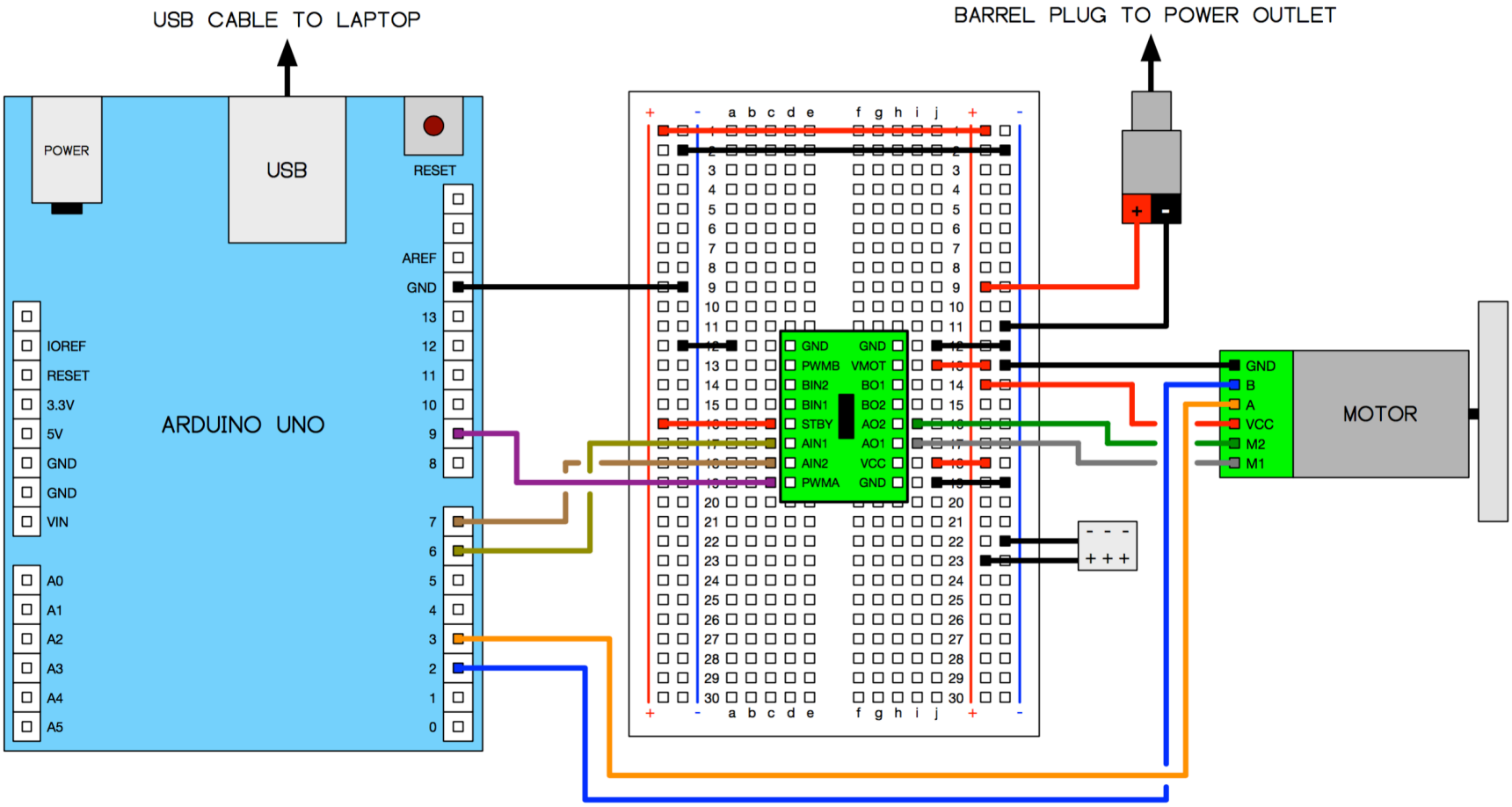


P = 30, I = 70, D = 1



DEMO!

Hardware


USB CABLE TO LAPTOP
BARREL PLUG TO POWER OUTLET
POWER
USB
RESET
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
1
0
IOREF
RESET
3.3V
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
ARDUINO UNO
GND
PWMB
BIN2
BIN1
STBY
AIN1
AIN2
PWMA
GND
VMOT
BO1
BO2
AO2
AO1
VCC
GND
GND
B
A
VCC
M2
M1
MOTOR