```
from DRV8833 import *
 1
 2
   from Encoder import *
 3
   class MotorController:
4
5
 6
        def __init__(self, motor, encoder):
            '''Controller for a single motor
7
            motor: motor driver (DRV8833)
8
            encoder: motor encoder (Encoder)
9
10
            self.mot = motor
11
12
            self.end = encoder
13
            self.integ = 0
14
        def p_control(self, desired_cps, P=1):
15
            '''Set motor control to rotate at desired_cps'''
16
            actual_cps = self.end.get_cps()
17
            error = desired cps - actual cps
18
            self.mot.set_speed(P*error)
19
20
            # return speed (e.g. for plotting)
            return actual_cps
21
22
23
        # add new method:
        def pi_control(self, desired_cps, Ts, P=1, I=1):
24
25
            actual cps = self.end.get cps()
26
            error = desired_cps - actual_cps
27
            self.integ += error * Ts
            # clamp integrator, e.g. if desired_cps exceeds maximum motor speed
28
            self.integ = max(-150, min(self.integ, 150))
29
            self.mot.set speed(P*error + I*self.integ)
30
            return actual cps
31
```

```
from DRV8833 import *
 1
   from Encoder import *
 2
 3
4
   class MotorController:
5
        def __init__(self, motor, encoder):
 6
            '''Controller for a single motor
7
            motor: motor driver (DRV8833)
8
9
            encoder: motor encoder (Encoder)
10
            self.mot = motor
11
            self.end = encoder
12
13
        def p control(self, desired cps, P=1):
14
15
            '''Set motor control to rotate at desired_cps'''
            actual_cps = self.end.get_cps()
16
            error = desired_cps - actual_cps
17
            self.mot.set_speed(P*error)
18
19
            # return speed (e.g. for plotting)
20
            return actual_cps
```

4/15/2019 p_controller.py

```
from MotorController import *
 1
   from DRV8833 import *
 2
   from Encoder import *
 3
4
5
   desired_cps = 100 # controller setpoint
                       # controller proportional gain
   P = 1
6
7
   Ts = 20
                        # controller operating period in [ms]
8
9
   controller = MotorController(DRV8833(19, 16), Encoder(34, 39, 0))
10
   def callback(timer):
11
        global controller, desired_cps, P
12
        # proportional control and print actual_cps (for plotting)
13
        print(controller.p_control(desired_cps, P))
14
15
   timer = Timer(0)
16
   timer.init(period=Ts, mode=Timer.UP, callback=callback(timer))
17
```

4/15/2019 pi_controller.py

```
from MotorController import *
 1
   from DRV8833 import *
 2
   from Encoder import *
 3
4
5
   desired_cps = 100 # controller setpoint
                       # controller proportional gain
   P = 1
6
7
   Ts = 20
                        # controller operating period in [ms]
8
9
   controller = MotorController(DRV8833(19, 16), Encoder(34, 39, 0))
10
   def callback(timer):
11
        global controller, desired_cps, P
12
        # proportional control and print actual_cps (for plotting)
13
        print(controller.pi control(desired cps, P))
14
15
   timer = Timer(0)
16
   timer.init(period=Ts, mode=Timer.UP, callback=callback(timer))
17
```