```
from machine import Pin, PWM

class DRV8833:

    def __init__(self, pinA, pinB, frequency=10000):
        '''Instantiate controller for one motor.
        pinA: pin connected to AIN1 or BIN1
        pinB: pin connected to AIN2 or BIN2
        frequency: pwm frequency
        '''
        self.pin1 = PWM(Pin(pinA), freq=frequency, timer=2)
        self.pin2 = PWM(Pin(pinB), freq=frequency, timer=3)

    def set_speed(self, value):
        '''value: -100 ... 100 sets speed (duty cycle) and direction'''
        if value > 0:
            self.pin1.duty(100)
            self.pin2.duty(value)
        else if value < 0:
            self.pin1.duty(value)
            self.pin2.duty(100)
        else:
            self.pin1.duty(value)
            self.pin2.duty(value)

```

```
from machine import Pin
from machine import DEC

class Encoder:

    def __init__(self, chA, chB, unit, counts_per_turn=24*75, wheel_diameter=330):
        '''Decode output from quadrature encoder connected to pins chA, chB.
        unit: DEC unit to use (0 ... 7).
        counts_per_turn: Number of counts per turn of the motor drive shaft. For scaling cps to rpm.
        wheel_diameter: In [mm]. For scaling count to distance traveled.
        '''
        self.p1 = Pin(chA, mode=Pin.IN, ...)
        self.p2 = Pin(chB, mode=Pin.IN, ...)
        self.cpt = counts_per_turn
        self.wd = wheel_diameter
        self.dec = DEC(unit, p1, p2)
        self.count = 0
        self.time = 0
        self.cps = 0

    def get_count(self):
        return dec.count()

    def get_distance(self):
        '''Distance traveled in [m].'''
        return get_count() * 3.14 * self.wd / 1000

    def get_cps(self):
        if self.count = 0:
            self.count = dec.count()
            self.time = time.clock()
            return 0
        else:
            count = dec.count()
            curr_time = time.clock()
            diff = count - self.count
            timediff = self.time - curr_time
            self.time = curr_time
            self.count = dec.count()
            self.cps = diff/timediff
            return self.cps

    def get_rpm(self):
        return self.cps/self.cpt

# Example:
import time

encA = Encoder(34, 39, 0)
```