```python
#!/usr/bin/python

from Raspi_PWM_Servo_Driver import PWM
import time

class Raspi_StepperMotor:
    MICROSTEPS = 8
    MICROSTEP_CURVE = [0, 50, 98, 142, 180, 212, 236, 250, 255]

    #MICROSTEPS = 16
    # a sinusoidal curve NOT LINEAR!
    #MICROSTEP_CURVE = [0, 25, 50, 74, 98, 120, 141, 162, 180, 197, 212, 225, 236, 244,
    250, 253, 255]

    def __init__(self, controller, num, steps=200):
        self.MC = controller
        self.revsteps = steps
        self.motornum = num
        self.sec_per_step = 0.1
        self.steppingcounter = 0
        self.currentstep = 0

        num -= 1

        if (num == 0):
            self.PWMA = 8
            self.AIN2 = 9
            self.AIN1 = 10
            self.PWMB = 13
            self.BIN2 = 12
            self.BIN1 = 11
        elif (num == 1):
            self.PWMA = 2
            self.AIN2 = 3
            self.AIN1 = 4
            self.PWMB = 7
            self.BIN2 = 6
            self.BIN1 = 5
        else:
            raise NameError('MotorHAT Stepper must be between 1 and 2 inclusive')

    def setSpeed(self, rpm):
        self.sec_per_step = 60.0 / (self.revsteps * rpm)
        self.steppingcounter = 0

    def oneStep(self, dir, style):
        pwm_a = pwm_b = 255

        # first determine what sort of stepping procedure we're up to
        if (style == Raspi_MotorHAT.SINGLE):
            if ((self.currentstep/(self.MICROSTEPS/2)) % 2):
                # we're at an odd step, weird
                if (dir == Raspi_MotorHAT.FORWARD):
                    self.currentstep += self.MICROSTEPS/2
                else:
                    self.currentstep -= self.MICROSTEPS/2
            else:
                # go to next even step
                if (dir == Raspi_MotorHAT.FORWARD):
                    self.currentstep += self.MICROSTEPS
                else:
                    self.currentstep -= self.MICROSTEPS
        if (style == Raspi_MotorHAT.DOUBLE):
            if not (self.currentstep/(self.MICROSTEPS/2) % 2):
                # we're at an even step, weird
                if (dir == Raspi_MotorHAT.FORWARD):
                    self.currentstep += self.MICROSTEPS/2
```

```
67                  else:
68                      self.currentstep -= self.MICROSTEPS/2
69              else:
70                  # go to next odd step
71                  if (dir == Raspi_MotorHAT.FORWARD):
72                      self.currentstep += self.MICROSTEPS
73                  else:
74                      self.currentstep -= self.MICROSTEPS
75          if (style == Raspi_MotorHAT.INTERLEAVE):
76              if (dir == Raspi_MotorHAT.FORWARD):
77                  self.currentstep += self.MICROSTEPS/2
78              else:
79                  self.currentstep -= self.MICROSTEPS/2
80
81          if (style == Raspi_MotorHAT.MICROSTEP):
82              if (dir == Raspi_MotorHAT.FORWARD):
83                  self.currentstep += 1
84              else:
85                  self.currentstep -= 1
86
87                  # go to next 'step' and wrap around
88                  self.currentstep += self.MICROSTEPS * 4
89                  self.currentstep %= self.MICROSTEPS * 4
90
91                  pwm_a = pwm_b = 0
92              if (self.currentstep >= 0) and (self.currentstep < self.MICROSTEPS):
93                  pwm_a = self.MICROSTEP_CURVE[self.MICROSTEPS - self.currentstep]
94                  pwm_b = self.MICROSTEP_CURVE[self.currentstep]
95              elif (self.currentstep >= self.MICROSTEPS) and (self.currentstep <
                self.MICROSTEPS*2):
96                  pwm_a = self.MICROSTEP_CURVE[self.currentstep - self.MICROSTEPS]
97                  pwm_b = self.MICROSTEP_CURVE[self.MICROSTEPS*2 - self.currentstep]
98              elif (self.currentstep >= self.MICROSTEPS*2) and (self.currentstep <
                self.MICROSTEPS*3):
99                  pwm_a = self.MICROSTEP_CURVE[self.MICROSTEPS*3 - self.currentstep]
100                 pwm_b = self.MICROSTEP_CURVE[self.currentstep - self.MICROSTEPS*2]
101             elif (self.currentstep >= self.MICROSTEPS*3) and (self.currentstep <
                self.MICROSTEPS*4):
102                 pwm_a = self.MICROSTEP_CURVE[self.currentstep - self.MICROSTEPS*3]
103                 pwm_b = self.MICROSTEP_CURVE[self.MICROSTEPS*4 - self.currentstep]
104
105
106         # go to next 'step' and wrap around
107         self.currentstep += self.MICROSTEPS * 4
108         self.currentstep %= self.MICROSTEPS * 4
109
110         # only really used for microstepping, otherwise always on!
111         self.MC._pwm.setPWM(self.PWMA, 0, pwm_a*16)
112         self.MC._pwm.setPWM(self.PWMB, 0, pwm_b*16)
113
114         # set up coil energizing!
115         coils = [0, 0, 0, 0]
116
117         if (style == Raspi_MotorHAT.MICROSTEP):
118             if (self.currentstep >= 0) and (self.currentstep < self.MICROSTEPS):
119                 coils = [1, 1, 0, 0]
120
121             elif(self.currentstep>=self.MICROSTEPS)and(self.currentstep<self.MICROSTEPS*2
                ):
122                 coils = [0, 1, 1, 0]
123
124             elif(self.currentstep>=self.MICROSTEPS*2)and(self.currentstep<self.MICROSTEPS
                *3):
125                 coils = [0, 0, 1, 1]
126
127             elif(self.currentstep>=self.MICROSTEPS*3)and(self.currentstep<self.MICROSTEPS
                *4):
```

```
                coils = [1, 0, 0, 1]
        else:
            step2coils = [  [1, 0, 0, 0],
                [1, 1, 0, 0],
                [0, 1, 0, 0],
                [0, 1, 1, 0],
                [0, 0, 1, 0],
                [0, 0, 1, 1],
                [0, 0, 0, 1],
                [1, 0, 0, 1] ]
            coils = step2coils[self.currentstep/(self.MICROSTEPS/2)]

        #print "coils state = " + str(coils)
        self.MC.setPin(self.AIN2, coils[0])
        self.MC.setPin(self.BIN1, coils[1])
        self.MC.setPin(self.AIN1, coils[2])
        self.MC.setPin(self.BIN2, coils[3])

        return self.currentstep

    def step(self, steps, direction, stepstyle):
        s_per_s = self.sec_per_step
        lateststep = 0

        if (stepstyle == Raspi_MotorHAT.INTERLEAVE):
            s_per_s = s_per_s / 2.0
        if (stepstyle == Raspi_MotorHAT.MICROSTEP):
            s_per_s /= self.MICROSTEPS
            steps *= self.MICROSTEPS

        print(s_per_s, " sec per step")

        for s in range(steps):
            lateststep = self.oneStep(direction, stepstyle)
            time.sleep(s_per_s)

        if (stepstyle == Raspi_MotorHAT.MICROSTEP):
            # this is an edge case, if we are in between full steps, lets just keep going
            # so we end on a full step
            while (lateststep != 0) and (lateststep != self.MICROSTEPS):
                lateststep = self.oneStep(dir, stepstyle)
                time.sleep(s_per_s)

class Raspi_DCMotor:
    def __init__(self, controller, num):
        self.MC = controller
        self.motornum = num
        pwm=0
        in1=0
        in2=0

        if(num==0):
                pwm = 8
                in2 = 9
                in1 = 10
        elif (num == 1):
                pwm=13
                in2=12
                in1=11
        elif(num==2):
                pwm=2
                in2=3
                in1=4
        elif(num==3):
                pwm=7
                in2=6
                in1=5
```

```
192             else:
193                 raise NameError('MotorHAT Motor must be between 1 and 4 inclusive')
194             self.PWMpin=pwm
195             self.IN1pin=in1
196             self.IN2pin=in2
197 
198         def run(self, command):
199             if not self.MC:
200                 return
201             if (command == Raspi_MotorHAT.FORWARD):
202                 self.MC.setPin(self.IN2pin, 0)
203                 self.MC.setPin(self.IN1pin, 1)
204             if (command == Raspi_MotorHAT.BACKWARD):
205                 self.MC.setPin(self.IN1pin, 0)
206                 self.MC.setPin(self.IN2pin, 1)
207             if (command == Raspi_MotorHAT.RELEASE):
208                 self.MC.setPin(self.IN1pin, 0)
209                 self.MC.setPin(self.IN2pin, 0)
210         def setSpeed(self, speed):
211             if (speed < 0):
212                 speed = 0
213             if (speed > 255):
214                 speed = 255
215             self.MC._pwm.setPWM(self.PWMpin, 0, speed*16)
216 
217     class Raspi_MotorHAT:
218         FORWARD = 1
219         BACKWARD = 2
220         BRAKE = 3
221         RELEASE = 4
222 
223         SINGLE = 1
224         DOUBLE = 2
225         INTERLEAVE = 3
226         MICROSTEP = 4
227 
228         def __init__(self, addr = 0x60, freq = 1600):
229             self._i2caddr = addr            # default addr on HAT
230             self._frequency = freq      # default @1600Hz PWM freq
231             self.motors = [ Raspi_DCMotor(self, m) for m in range(4) ]
232             self.steppers = [ Raspi_StepperMotor(self, 1), Raspi_StepperMotor(self, 2) ]
233             self._pwm =  PWM(addr, debug=False)
234             self._pwm.setPWMFreq(self._frequency)
235 
236         def setPin(self, pin, value):
237             if (pin < 0) or (pin > 15):
238                 raise NameError('PWM pin must be between 0 and 15 inclusive')
239             if (value != 0) and (value != 1):
240                 raise NameError('Pin value must be 0 or 1!')
241             if (value == 0):
242                 self._pwm.setPWM(pin, 0, 4096)
243             if (value == 1):
244                 self._pwm.setPWM(pin, 4096, 0)
245 
246         def getStepper(self, steps, num):
247             if (num < 1) or (num > 2):
248                 raiseNameError('MotorHATSteppermustbebetween1and2inclusive')
249             return self.steppers[num-1]
250 
251         def getMotor(self, num):
252             if (num < 1) or (num > 4):
253                 raise NameError('MotorHAT Motor must be between 1 and 4 inclusive')
254             return self.motors[num-1]
255 
```