```python
#!/usr/bin/env python
"""
   pid_velocity - takes messages on wheel_vtarget
      target velocities for the wheels and monitors wheel for feedback

    Copyright (C) 2012 Jon Stephan.

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

import rospy
import roslib

from std_msgs.msg import Int32 as Int16
from std_msgs.msg import Int8
from std_msgs.msg import Float32
from numpy import array


########################################################
########################################################
class PidVelocity():
########################################################
########################################################


    ########################################################
    def __init__(self):
    ########################################################
        rospy.init_node("pid_velocity")
        self.nodename = rospy.get_name()
        rospy.loginfo("%s started" % self.nodename)

        ### initialize variables
        self.target = 0
        self.motor = 0
        self.vel = 0
        self.integral = 0
        self.error = 0
        self.derivative = 0
        self.previous_error = 0
        self.wheel_prev = 0
        self.wheel_latest = 0
        self.then = rospy.Time.now()
        self.wheel_mult = 0
        self.prev_encoder = 0

        ### get parameters ####
        self.Kp = rospy.get_param('~Kp',10)
```

```
61          self.Ki = rospy.get_param('~Ki',10)
62          self.Kd = rospy.get_param('~Kd',0.001)
63          self.out_min = rospy.get_param('~out_min',-255)
64          self.out_max = rospy.get_param('~out_max',255)
65          self.rate = rospy.get_param('~rate',30)
66          self.max_change = rospy.get_param('~max_change',0.666666)
67          self.timeout_ticks = rospy.get_param('~timeout_ticks',4)
68          self.ticks_per_meter = rospy.get_param('ticks_meter', 20)
69          self.vel_threshold = rospy.get_param('~vel_threshold', 0.001)
70          self.encoder_min = rospy.get_param('encoder_min', -32768)
71          self.encoder_max = rospy.get_param('encoder_max', 32768)
72          self.encoder_low_wrap = rospy.get_param('wheel_low_wrap', (self.encoder_max
    - self.encoder_min) * 0.3 + self.encoder_min )
73          self.encoder_high_wrap = rospy.get_param('wheel_high_wrap',
    (self.encoder_max - self.encoder_min) * 0.7 + self.encoder_min )
74          self.wheel_latest = 0.0
75          self.prev_pid_time = rospy.Time.now()
76          rospy.logdebug("%s got Kp:%0.3f Ki:%0.3f Kd:%0.3f tpm:%0.3f" %
    (self.nodename, self.Kp, self.Ki, self.Kd, self.ticks_per_meter))
77
78          #### subscribers/publishers
79          rospy.Subscriber("wheel", Int16, self.wheelCallback)
80          rospy.Subscriber("wheel_vtarget", Float32, self.targetCallback)
81          self.pub_motor = rospy.Publisher('motor_cmd',Int8, queue_size=10)
82          self.pub_vel = rospy.Publisher('wheel_vel', Float32, queue_size=10)
83
84
85      ##########################################################
86      def spin(self):
87      ##########################################################
88          self.r = rospy.Rate(self.rate)
89          self.then = rospy.Time.now()
90          self.ticks_since_target = self.timeout_ticks
91          self.wheel_prev = self.wheel_latest
92          self.then = rospy.Time.now()
93          while not rospy.is_shutdown():
94              self.spinOnce()
95              self.r.sleep()
96
97      ##########################################################
98      def spinOnce(self):
99      ##########################################################
100         self.previous_error = 0.0
101         self.integral = 0.0
102         self.error = 0.0
103         self.derivative = 0.0
104         self.vel = 0.0
105   self.motor = 0
106         # only do the loop if we've recently recieved a target velocity message
107         while not rospy.is_shutdown() and self.ticks_since_target <
    self.timeout_ticks:
108             self.calcVelocity()
109             self.doPid()
110             self.pub_motor.publish(self.motor)
111             self.r.sleep()
112             self.ticks_since_target += 1
113             if self.ticks_since_target == self.timeout_ticks:
114                 self.pub_motor.publish(0)
115     rospy.loginfo("timeout")
116
```

```
117     ########################################################
118     def calcVelocity(self):
119     ########################################################
120         self.dt_duration = rospy.Time.now() - self.then
121         self.dt = self.dt_duration.to_sec()
122         rospy.logdebug("-D- %s caclVelocity dt=%0.3f wheel_latest=%0.3f
    wheel_prev=%0.3f" % (self.nodename, self.dt, self.wheel_latest, self.wheel_prev))
123 
124         # we received a new wheel value
125   if(self.dt != 0):
126           cur_vel = (self.wheel_latest - self.wheel_prev) / self.dt
127     self.updateVel(cur_vel)
128           rospy.logdebug("-D- %s **** wheel updated vel=%0.3f **** " %
    (self.nodename, self.vel))
129           self.wheel_prev = self.wheel_latest
130           self.then = rospy.Time.now()
131     self.pub_vel.publish(self.vel)
132         else:
133     rospy.loginfo("dt was zero")
134     ########################################################
135     def updateVel(self, val):
136     ########################################################
137         self.vel = self.vel * 0.8 + 0.2 * val
138     def updateDerv(self,val):
139         self.derivative = val * 0.8 + self.derivative * 0.2
140     ########################################################
141     def doPid(self):
142     ########################################################
143         pid_dt = self.dt
144   if(self.dt!=0):
145           self.prev_pid_time = rospy.Time.now()
146 
147           self.error = self.target - self.vel
148           self.integral = self.integral + (self.error * pid_dt)
149         # rospy.loginfo("i = i + (e * dt):  %0.3f = %0.3f + (%0.3f * %0.3f)" %
    (self.integral, self.integral, self.error, pid_dt))
150           self.updateDerv((self.error - self.previous_error) / pid_dt)
151           self.previous_error = self.error
152         #Make PID loop a passthrough
153     #self.motor = self.target
154           self.motor += max(min((self.Kp * self.error) + (self.Ki * self.integral) +
    (self.Kd * self.derivative),self.max_change),-self.max_change)
155           self.motor = min(max(self.motor,self.out_min),self.out_max)
156 
157         #if (self.target == 0):
158         #    self.motor = 0
159 
160           rospy.logdebug("vel:%0.2f tar:%0.2f err:%0.2f int:%0.2f der:%0.2f ##
    motor:%d " %
161                          (self.vel, self.target, self.error, self.integral,
    self.derivative, self.motor))
162       else:
163     rospy.loginfo("dt was zero")
164 
165     ########################################################
166     def wheelCallback(self, msg):
167     ########################################################
168         enc = msg.data
169         if (enc < self.encoder_low_wrap and self.prev_encoder >
    self.encoder_high_wrap) :
```

```
170             self.wheel_mult = self.wheel_mult + 1
171
172         if (enc > self.encoder_high_wrap and self.prev_encoder <
    self.encoder_low_wrap) :
173             self.wheel_mult = self.wheel_mult - 1
174
175
176         self.wheel_latest = 1.0 * (enc + self.wheel_mult * (self.encoder_max -
    self.encoder_min)) / self.ticks_per_meter
177         self.prev_encoder = enc
178
179
180 #         rospy.logdebug("-D- %s wheelCallback msg.data= %0.3f wheel_latest = %0.3f
    mult=%0.3f" % (self.nodename, enc, self.wheel_latest, self.wheel_mult))
181
182     ############################################################
183     def targetCallback(self, msg):
184     ############################################################
185         self.target = msg.data
186         self.ticks_since_target = 0
187         # rospy.logdebug("-D- %s targetCallback " % (self.nodename))
188
189
190 if __name__ == '__main__':
191     """ main """
192     try:
193         pidVelocity = PidVelocity()
194         pidVelocity.spin()
195     except rospy.ROSInterruptException:
196         pass
197
```