ДОДАТОК А

Avoid.py:

```
from Ultrasonic import ultrasonic
from picar import front_wheels
from picar import back_wheels
import time
import picar
import random

force_turning = 0    # 0 = random direction, 1 = force left, 2 = force right, 3 =
orderdly

picar.setup()

ua = Ultrasonic_Avoidance.Ultrasonic_Avoidance(20)
fw = front_wheels.Front_Wheels(db='config')
bw = back_wheels.Back_Wheels(db='config')
fw.turning_max = 45

forward_speed = 70
backward_speed = 70

back_distance = 10
turn_distance = 20
```

```
timeout = 10
last_angle = 90
last_dir = 0
def rand_dir():
    global last_angle, last_dir
    if force_turning == 0:
        _dir = random.randint(0, 1)
    elif force_turning == 3:
        _dir = not last_dir
        last_dir = _dir
        print('last dir  %s' % last_dir)
    else:
        _dir = force_turning - 1
    angle = (90 - fw.turning_max) + (_dir * 2* fw.turning_max)
    last_angle = angle
    return angle

def opposite_angle():
    global last_angle
    if last_angle < 90:
        angle = last_angle + 2* fw.turning_max
    else:
        angle = last_angle - 2* fw.turning_max
    last_angle = angle
    return angle
```

```
def start_avoidance():
    print('start_avoidance')

    count = 0
    while True:
        distance = ua.get_distance()
        print("distance: %scm" % distance)
        if distance > 0:
            count = 0
            if distance < back_distance: # backward
                print( "backward")
                fw.turn(opposite_angle())
                bw.backward()
                bw.speed = backward_speed
                time.sleep(1)
                fw.turn(opposite_angle())
                bw.forward()
                time.sleep(1)
            elif distance < turn_distance: # turn
                print("turn")
                fw.turn(rand_dir())
                bw.forward()
                bw.speed = forward_speed
                time.sleep(1)
            else:
                fw.turn_straight()
                bw.forward()
                bw.speed = forward_speed

        else:                           # forward
            fw.turn_straight()
            if count > timeout:  # timeout, stop;
                bw.stop()
            else:
                bw.backward()
                bw.speed = forward_speed
                count += 1

def stop():
    bw.stop()
    fw.turn_straight()

if __name__ == '__main__':
    try:
        start_avoidance()
    except KeyboardInterrupt:
        stop()
```

ultrasonic.py

```
import time
import RPi.GPIO as GPIO

class Ultrasonic_Avoidance(object):
    timeout = 0.05

    def __init__(self, channel):
        self.channel = channel
        GPIO.setmode(GPIO.BCM)

    def distance(self):
        pulse_end = 0
        pulse_start = 0
        GPIO.setup(self.channel,GPIO.OUT)
        GPIO.output(self.channel, False)
        time.sleep(0.01)
        GPIO.output(self.channel, True)
        time.sleep(0.00001)
        GPIO.output(self.channel, False)
        GPIO.setup(self.channel,GPIO.IN)

        timeout_start = time.time()
        while GPIO.input(self.channel)==0:
            pulse_start = time.time()
            if pulse_start - timeout_start > self.timeout:
                return -1
        while GPIO.input(self.channel)==1:
            pulse_end = time.time()
            if pulse_start - timeout_start > self.timeout:
                return -1

        if pulse_start != 0 and pulse_end != 0:
            pulse_duration = pulse_end - pulse_start
            distance = pulse_duration * 100 * 343.0 /2
            distance = int(distance)
            #print('start = %s'%pulse_start,)
            #print('end = %s'%pulse_end)
            if distance >= 0:
                return distance
            else:
                return -1
        else :
            #print('start = %s'%pulse_start,)
            #print('end = %s'%pulse_end)
            return -1

    def get_distance(self, mount = 5):
        sum = 0
        for i in range(mount):
            a = self.distance()
            #print('    %s' % a)
```

```
                sum += a
            return int(sum/mount)
        def less_than(self, alarm_gate):
            dis = self.get_distance()
            status = 0
            if dis >=0 and dis <= alarm_gate:
                status = 1
            elif dis > alarm_gate:
                status = 0
            else:
                status = -1
            #print('distance =',dis)
            #print('status =',status)
            return status

    def test():
        UA = Ultrasonic_Avoidance(17)
        threshold = 10
        while True:
            distance = UA.get_distance()
            status = UA.less_than(threshold)
            if distance != -1:
                print('distance', distance, 'cm')
                time.sleep(0.2)
            else:
```

```
            print(False)
        if status == 1:
            print("Less than %d" % threshold)
        elif status == 0:
            print("Over %d" % threshold)
        else:
            print("Read distance error.")

if __name__ == '__main__':
    test()
```