```
# -*- coding: utf-8 -*-
##### Suggested clean drone startup sequence #####
import time, sys
import ps_drone                 #Imports the PS-Drone-API

drone = ps_drone.Drone()        #Initials the PS-Drone-API
drone.startup()                 #Connects to the drone and starts subprocesses

drone.reset()                   #Sets drone’s LEDs to green when red

while (drone.getBattery()[0]==-1):
    time.sleep(0.1)             #Reset completed

print "Battery: " + str(drone.getBattery()[0]) + "%" + str(drone.getBattery()[1])
if drone.getBattery()[1]== "empty": #Check Battery status
     sys.exit()

drone.useDemoMode(True)         #Use demo mode
drone.getNDpackage(["demo", "altitude", "vision_detect"])    #Packets, which shall be decoded
time.sleep(0.5)                 #Give it some time to fully awake



                    ###### Main Program ######
drone.takeoff()
while drone.NavData["demo"][0][2]:
    time.sleep(0.1) #Still in landed-mode

gliding = False
time.sleep(1)                 #Gives the drone time to

xStart = time.time()        # Start timer for x
xEnd = time.time()          # End timer for x
x = 0                       # Set x = 0, this will be reset later for the Distance
y = 0                       # Set y = 0, this will be reset later for the Distance

# Setting up detection...
# Oriented Roundel=128
drone.setConfig("detect:detect_type", "10")                  # 10 = Enable universal detection
drone.setConfig("detect:detections_select_h", "0")           # Turn off detection for front
camera
drone.setConfig("detect:detections_select_v", "128")         # Detect "Oriented Roundel"
with ground-camera
CDC = drone.ConfigDataCount
while CDC == drone.ConfigDataCount:    time.sleep(0.01)        # Wait until configuration
has been set




# tagDetected function that will be called if a tag has been detected
def tagDetected ():
    for i in range (0,tagNum):
        # Print the tag number and coordinates
        print "Tag no "+str(i)+" : X= "+str(tagX[i])+"  Y= "+str(tagY[i])+"  Dist= "+str(tagZ
        [i])+"  Orientation= "+str(tagRot[i])
        # If the tag is closer than 300mm the drone will land
        if (tagZ[i] <= 300):
            drone.stop()                    #Drone stops...
            drone.land()
            sys.exit()


NDC = drone.NavDataCount
while NDC == drone.NavDataCount:   time.sleep(0.01)

# Drone flies forward at a speed of 0.2
drone.moveForward(0.2)

while (xEnd - xStart < 4):       #While loop that lasts 4 seconds

    tagNum = drone.NavData["vision_detect"][0]                 # Number of found tags
    tagX =   drone.NavData["vision_detect"][2]                 # Horizontal position(s)
    tagY =   drone.NavData["vision_detect"][3]                 # Vertical position(s)
    tagZ =   drone.NavData["vision_detect"][6]                 # Distance(s)
    tagRot = drone.NavData["vision_detect"][7]                 # Orientation(s)

    print "Estimated speed in X in mm/s: " +str(drone.NavData['demo'][4][0]) # Estimated
    speed in X in mm/s
    speed = drone.NavData['demo'][4][0]
    fifth = speed/5                 # Divide speed by 5 to allow for acceleration at a fifth
    of a second
    x = x + (fifth * 0.2)           # Calculate the distance every 0.2 of a second and add
    to value x
    time.sleep(0.2)                 # Print this value every 0.2 second(s)
    xEnd = time.time()              # End value is reset every iteration to get the length
    of flight
    if tagNum:                      # if a tag is detected, the tagDetected() method is called
        tagDetected()
# Convert x to metres
x = x/100

# x distance is printed on the screen and formatted to 2 decimals
print "\nDistance: " + str('{0:.3g}'.format(x)) + " Metres"

drone.stop()                    # Drone stops
time.sleep(1)                   # Given time to stop
drone.turnAngle(90,1)           # Drone turns right 90 degrees at full speed
drone.stop()                    # Drone stops
time.sleep(1)                 # Given time to stop

yStart = time.time()            # y timer is started
yEnd = time.time()              # y end timer for when the timer is finished

drone.moveForward(0.2)          # Drone moves forward at a speed of 0.2
while (yEnd - yStart < 2.5):    # While loop that lasts 2.5 seconds

    tagNum = drone.NavData["vision_detect"][0]                 # Number of found tags
    tagX =   drone.NavData["vision_detect"][2]                 # Horizontal position(s)
    tagY =   drone.NavData["vision_detect"][3]                 # Vertical position(s)
    tagZ =   drone.NavData["vision_detect"][6]                 # Distance(s)
    tagRot = drone.NavData["vision_detect"][7]                 # Orientation(s)

    print "Estimated speed in X in mm/s: " +str(drone.NavData['demo'][4][0]) # Estimated
    speed in X in mm/s
    speed = drone.NavData['demo'][4][0]
    fifth = speed/5               # Divide speed by 5 to allow for acceleration at a fifth
    of a second
    y = y + (fifth * 0.2)         # Calculate the distance every 0.2 of a second and add to
    value y
    time.sleep(0.2)               # Print this value every 0.2 second(s)
    yEnd = time.time()            # End value is reset every iteration to get the length of
    flight

    if tagNum:                    # Call tagDetected() method if a tag is found
        tagDetected()


# Covert y to Metres
y = y/100

# y distance is printed on the screen and formatted to 2 decimals
print "\nDistance: " + str('{0:.3g}'.format(y)) + " Metres"

area = x * y                    # Calculate area

# Print the area result on the screen
# Format the area to two decimal places
print "Area: " + str('{0:.3g}'.format(area)) + " Metres Squared"



###### Complete the square and land #######




drone.stop()                    # Drone stops...
time.sleep(2)                   # time to stop
drone.turnAngle(70,1)           # Drone turns right 70 degrees at full speed
drone.stop()                    # Drone stops
time.sleep(0.3)                 # time to stop

xStart = time.time()            # Timer is used again and reset to the present time
xEnd = time.time()              # End timer is reset again

drone.moveForward(0.2)          # Drone moves forward at 0.2 speed
while (xEnd - xStart < 2.5):    # While loop that lasts 2.5 seconds

    tagNum = drone.NavData["vision_detect"][0]                 # Number of found tags
    tagX =   drone.NavData["vision_detect"][2]                 # Horizontal position(s)
    tagY =   drone.NavData["vision_detect"][3]                 # Vertical position(s)
    tagZ =   drone.NavData["vision_detect"][6]                 # Distance(s)
    tagRot = drone.NavData["vision_detect"][7]                 # Orientation(s)

    time.sleep(0.1)              # Every 0.1 seconds
    xEnd = time.time()           # End value is reset every iteration to get the length of
    flight

    if tagNum:
        tagDetected()
```

```
drone.stop()                    # Drone stops...
time.sleep(2)                   # Time to stop
drone.turnAngle(110,1)          # Drone turns right 110 degrees at full speed
drone.stop()                    # Drone stops
time.sleep(0.3)                 # Time to stop

yStart = time.time()            # Timer starts
yEnd = time.time()              # Timer end variable


drone.moveForward(0.2)          # Drone moves forward at 0.2 speed
while (yEnd - yStart < 2.5):    # While loop that lasts 2.5 seconds

    tagNum = drone.NavData["vision_detect"][0]                 # Number of found tags
    tagX =   drone.NavData["vision_detect"][2]                 # Horizontal position(s)
    tagY =   drone.NavData["vision_detect"][3]                 # Vertical position(s)
    tagZ =   drone.NavData["vision_detect"][6]                 # Distance(s)
    tagRot = drone.NavData["vision_detect"][7]                 # Orientation(s)

    time.sleep(0.1)              # Print this value every 1 second(s)
    yEnd = time.time()           # End value is reset every iteration to get the length of
    flight

    if tagNum:
        tagDetected()            # If a tag is detected the tagDetected() function is called



drone.stop()                    # Drone stops
time.sleep(2)                   # Time to stop
drone.turnAngle(110,1)          # Drone turns right 110 degrees at full speed
drone.stop()                    # Drone stops
time.sleep(0.5)                 # Time to stop


drone.land()                    #Drone lands

print 'Drone has landed.'

########### Program Finished ############
```