**Virtual Initio Programming: Work Sheets and Exercises 18 Sample Answers and Trouble Shooting**

**WS18**

**Challenge Problem Sample Answer:**

import simclient.simrobot as initio

import time

initio.init()

def drive\_to\_wall():

while (not initio.getDistance() < 20 and not initio.irLeft() and not initio.irRight()):

initio.forward(10)

initio.stop()

def spin(direction):

if (direction == 'right'):

initio.spinRight(10)

else:

initio.spinLeft(10)

def obstacle\_to(direction):

if (direction == 'right'):

return initio.irRight()

else:

return initio.irLeft()

def opposite\_direction(direction):

if (direction == 'right'):

return 'left'

else:

return 'right'

def follow\_wall(direction):

if (initio.getDistance() < 20):

while(initio.getDistance() < 20 or obstacle\_to(direction)):

spin(opposite\_direction(direction))

elif (obstacle\_to(direction)):

initio.forward(10)

time.sleep(2)

elif (not obstacle\_to(direction)):

initio.forward(10)

time.sleep(1)

spin(direction)

time.sleep(1)

drive\_to\_wall()

while (not initio.irLeftLine()):

follow\_wall('right')

initio.forward(10)

time.sleep(3)

initio.reverse(10)

time.sleep(10)

initio.spinLeft(10)

time.sleep(25)

while (not initio.irLeftLine()):

follow\_wall('left')

initio.stop()

**Troubleshooting:** It may take some experimentation to get sleep times correct. This above version of wall following can end up “bouncing along” the side of the wall since the robot becomes angled slightly towards it. Some fine tuning of timings might help with this. Really the robot needs more sensors at the side in order to help it determine the angle it is at with respect to the wall. So, the above is probably as good as can reasonably be expected.

**Ex18**

Sample Answer. Pretty much the same as above just without program commands at the end.

import simclient.simrobot as initio

import time

def drive\_to\_wall():

while (not initio.getDistance() < 20 and not initio.irLeft() and not initio.irRight()):

initio.forward(10)

initio.stop()

def spin(direction):

if (direction == 'right'):

initio.spinRight(10)

else:

initio.spinLeft(10)

def obstacle\_to(direction):

if (direction == 'right'):

return initio.irRight()

else:

return initio.irLeft()

def opposite\_direction(direction):

if (direction == 'right'):

return 'left'

else:

return 'right'

def follow\_wall(direction):

if (initio.getDistance() < 20):

while(initio.getDistance() < 20 or obstacle\_to(direction)):

spin(opposite\_direction(direction))

elif (obstacle\_to(direction)):

initio.forward(10)

time.sleep(2)

elif (not obstacle\_to(direction)):

initio.forward(10)

time.sleep(1)

spin(direction)

time.sleep(1)



University of Liverpool, 2019

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).