**Pi2Go Simulator Programming: Work Sheets and Exercises 18 Sample Answers and Trouble Shooting**

**WS18**

**Challenge Problem Sample Answer:**

**import** simclient.simrobot **as** pi2go

**import** time

pi2go.init()

**def** drive\_to\_wall():

**while** (**not** pi2go.irCentre() **and** **not** pi2go.irLeft() **and** **not** pi2go.irRight()):

pi2go.forward(10)

pi2go.stop()

**def** spin(direction):

**if** (direction == 'right'):

pi2go.spinRight(10)

**else**:

pi2go.spinLeft(10)

**def** obstacle\_to(direction):

**if** (direction == 'right'):

**return** pi2go.irRight()

**else**:

**return** pi2go.irLeft()

**def** opposite\_direction(direction):

**if** (direction == 'right'):

**return** 'left'

**else**:

**return** 'right'

**def** follow\_wall(direction):

**if** (pi2go.irCentre()):

**while**(pi2go.irCentre() **or** obstacle\_to(direction)):

spin(opposite\_direction(direction))

**elif** (obstacle\_to(direction)):

pi2go.forward(10)

time.sleep(2)

**elif** (**not** obstacle\_to(direction)):

spin(direction)

time.sleep(1)

drive\_to\_wall()

**while** (**not** pi2go.irLeftLine()):

follow\_wall('right')

pi2go.forward(10)

time.sleep(3)

pi2go.reverse(10)

time.sleep(10)

pi2go.spinLeft(10)

time.sleep(5)

**while** (**not** pi2go.irLeftLine()):

follow\_wall('left')

pi2go.stop()

**Troubleshooting:** It may take some experimentation to get sleep times correct. This above version of wall following often ends 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, as might using the ultra-sonic sensor as well as the irCentre sensor – but, to be honest, these steps make the program a lot more fiddly and don’t necessarily improve performance. 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. This has be tweaked a bit from the above in an attempt to reduce the “bouncing” effect. There’s no real right answer here since perfect performance isn’t possible without using quite sophisticated mathematical and statistical techniques in order to map the space.

import simclient.simrobot as pi2go

import time

def drive\_to\_wall():

while (not pi2go.irCentre() and not pi2go.irLeft() and not pi2go.irRight()):

pi2go.forward(10)

pi2go.stop()

def spin(direction):

if (direction == 'right'):

pi2go.spinRight(10)

else:

pi2go.spinLeft(10)

def obstacle\_to(direction):

if (direction == 'right'):

return pi2go.irRight()

else:

return pi2go.irLeft()

def opposite\_direction(direction):

if (direction == 'right'):

return 'left'

else:

return 'right'

def follow\_wall(direction):

while (not pi2go.getSwitch()):

if (pi2go.irCentre()):

while(pi2go.irCentre() or obstacle\_to(direction)):

spin(opposite\_direction(direction))

pi2go.forward(10)

time.sleep(2)

elif (obstacle\_to(direction)):

pi2go.forward(10)

time.sleep(2)

spin(opposite\_direction(direction))

time.sleep(2)

elif (not obstacle\_to(direction)):

pi2go.forward(10)

time.sleep(2)

spin(direction)

time.sleep(2)

pi2go.stop()



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/).