

CS 1301 — Homework 5– Avoiding Obstacles & Seeking Light

Due: T-Square turn in deadline: Wednesday, March 16th, before 6 PM

NOTE: You must demo your program in recitation on Wednesday! So it must be done before your recitation!

Scored out of 100 points
Files to submit: hw5.py

This is an Pair programming assignment!

You are

expected to work with the person you have been paired with in class, and you are both responsible for submitting the exact same code to T-Square. Your pair may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for other pairs. Collaboration at a reasonable level will not result in substantially similar code. For pair programming assignments, you and your partner should turn in identical assignments.

For Help:

- TA Helpdesk – Schedule posted on class website.
- Email TAs

Notes:

- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- **Do not wait until the last minute to do this assignment in case you run into problems.**
- If you find a significant error in the homework assignment, please let a TA know immediately.

Part one --- Avoid Walls Code – 35 points

There are five kinds of sensors on the robot: light sensor (detect the how bright the light is), proximity sensor (see whether there is anything around the robot), stall sensor, the camera, and the battery voltage. We are going to use the robot's proximity sensors for this homework. Be sure to read all of the information below. It explains many functions that are vital to the successful completion of this assignment.

Mission:

Your robot will be randomly placed in an arena of size 5 x 3 (Units: 11 in). You need to write a function called **avoidWalls()** to move your robot around for one minute (+/- 5 seconds) without hitting walls. The robot needs to be moving at a minimum of 1/3 of its maximum speed. (Your robot may drive “backwards” if you want to use the `getIR()` sensors instead of the `getObstacle()` sensors.) The robot should also celebrate after finishing the mission successfully. How it is going to celebrate is up to you, although it must be recognizable. We suggest moving around and beeping at a minimum.

For more information on the robot arena, see the posted file.

--- What's on the robot? ---

Proximity Sensors:

Proximity sensors are used to detect objects that are close to the robot. The robot has two sets of proximity sensors: one set is on the robot and the other set is on the fluke.

The IR sensors on the robot:

There are two infrared (IR) sensors on the back of the robot (Assuming the fluke is facing forwards). You use the sensors by calling the **getIR(<position>)** function.

Examples:

```
>>> getIR()
[1, 0]
>>> getIR('left')
1
>>> getIR(0)
1
>>> getIR('right')
0
>>> getIR(1)
0
```

getIR(<POSITION>) Returns an integer value in the <POSITION> IR sensor. <POSITION> can either be 'left' or 'right' or one of the numbers 0, 1, which correspond to “left”, and “right”.

IR sensors return either a 1 or a 0. A value of 1 implies that there is nothing in close proximity of the front of the sensor and a 0 implies that there is something right in front of it.

The proximity sensors on the Fluke:

There are three proximity sensors on the fluke: one front sensor and two side sensors. They give different values than the ones on the robot. To use this set of sensors, you need to call **getObstacle(<position>)**.

getObstacle() returns a list that contains the values from all three sensors. To get a value from a specific sensor, you can call **getObstacle(<position>)**. <position> can be “left”, “right” or “center”. <position> can also be number 0, 1, or 2, which correspond to “left”, “center”, and “right”.

Examples:

```
>>> getObstacle()
[1703, 1128, 142]
>>> getObstacle('left')
1703
>>> getObstacle(0)
1703
>>> getObstacle('center')
1128
>>> getObstacle(1)
1128
>>> getObstacle('right')
142
>>> getObstacle(2)
142
```

getObstacle(<position>) returns a integer value between 0 and 7000. A zero (0) indicates there is nothing in close proximity of the sensor. Higher value implies the presence of objects in front of the sensor(s). Note that the center sensor is the most accurate. The side sensors will detect objects located at about a 45 degree angle, but not those directly to the sides of the robot.

More details about the sensors can be found:

<http://wiki.roboteducation.org/Learning_Computing_With_Robots – Chapter 5 – Sensing the World>

Reminder: Robot needs to be moving at a minimum of 1/3 speed.

Internal Clock function:

To keep track of time, you can use the myro internal clock function. To get the current time, call `currentTime()` function. The function returns the number of seconds past since sometime in the past. [Epoch or Unix time if you’re interested.]

Example:

```
>>> currentTime()
1222374008.360949
```

To keep track of time, you need to call `currentTime()` and save the time to a variable (e.g. `time`). You can get the time that has passed since you last called `currentTime()` by subtracting time from `currentTime()`.

Example:

time = currentTime()

doing something.....

doing something.....

timePast = currentTime() - time

More detail about the internal clock function:

http://wiki.roboteducation.org/Learning_Computing_With_Robots – Chapter 4 – Sensing From Within

If you want, you may use the timeRemaining() function and a while loop instead of the currentTime() function to time your robots motion.

If you need help with the move functions, go to

http://wiki.roboteducation.org/Learning_Computing_With_Robots – Chapter 2 – Personal Robots

Turning it in:

Be sure to put the lines from myro import * and initialize() or init() at the beginning of the file (after the required comments). Be sure not to specify the port parameter in your initialize command, such as initialize(“com4”). This makes it very time consuming to grade if we have to go into your code and change the com port to the one that works on our specific system. The TA will type avoidWalls() to start your robot moving, so you don't need to include a call to that function in your homework file.

Reminder on collaboration statement:

This is a pair assignment. Please include your and your partners name at the top, and the names of any other people you collaborated with in your collaboration statement.

Part 2 : Seek Light - 25 points

For the second part of this assignment, you will create a function named seekLight(). When called, this function will look for the brightest source of light and move towards it. When testing, you should shine a flashlight at your robots light sensors from one or two feet away. The robot should turn towards the flashlight and approach it continuously for one minute. After the minute is up, your robot should stop and indicate that it is finished with a beep or some type of celebratory dance. (You can re-use the celebration you made for the avoidWalls part of the homework if you would like. You do not have to check for obstacles, assume that the person holding the flashlight will not lead your robot into any walls.)

You may use the getLight sensors on the back of the robot (driving backwards) or the getBright sensors built into the robots camera (driving forwards), your choice. Do not use sensors on each end of the robot, pick the front or the back.

Grading Criteria

Part 1 – Avoid Walls Grading Criteria: 35 points

File named correctly	5 pt
Uses iteration correctly	5pt
Uses IR obstacle sensors to detect obstacles (Walls)	10pt
Code to avoid walls!	5pts
Moves at 1/3 speed or higher for 1 minute	5pts
Celebration in the end	5pt

Part 2 - Seek Light Grading Criteria 25 points

Uses iteration correctly	5pt
Uses light sensors to detect brightest light	10pt
Code to turn/move towards brightest light!	5pts
Seeks light for 1 minute, then beeps or celebrates	5pts

**You will demo your robot in recitation, so make sure it's working before your recitation period!
Also note that EACH team member must understand how all the code works!**

Part 3 - Avoid Walls Demo Grading Criteria:

Moves around for 1 minute, doesn't hit walls:	10 pts
Randomly selected team member can explain how the code works:	10 pts

Part 4 - Seek Light Demo Grading Criteria:

Robot will follow a flashlight aimed at its light sensors:	10 pts
Randomly selected team member can explain how the code works:	10 pts