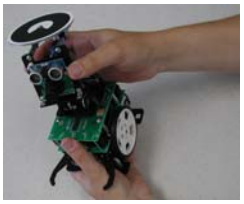
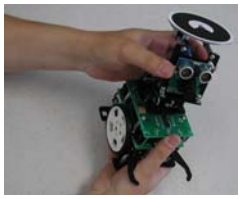


COMP4807 - LAB #3
Navigation
(Due Sun. Oct. 28, 2012 @ 11:59pm)



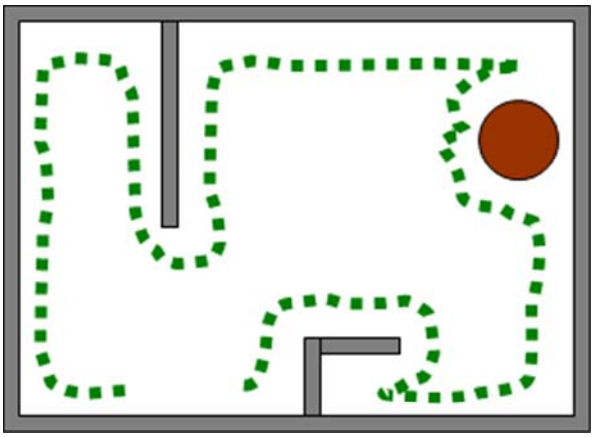
In this assignment you will make the robot perform wall-following and point to point travel using a Planner. As with the previous 2 assignments, it is a good idea to have some code already written BEFORE getting to the lab so that you can maximize the use of your lab time.

Robot Setup:

Make sure the robot is turned **off** and then turn **on** (i.e., down position) dip switches **1** (**F_IR**), **2** (**S_IR**) and **3** (**BLU**) from the 8-position switch on the top of the robot and turn **off** (i.e., up) all other switches (i.e., **4, 5, 6, 7** and **8**). At the back of the robot, make sure that the 2-position dipswitches are both off (i.e., up).

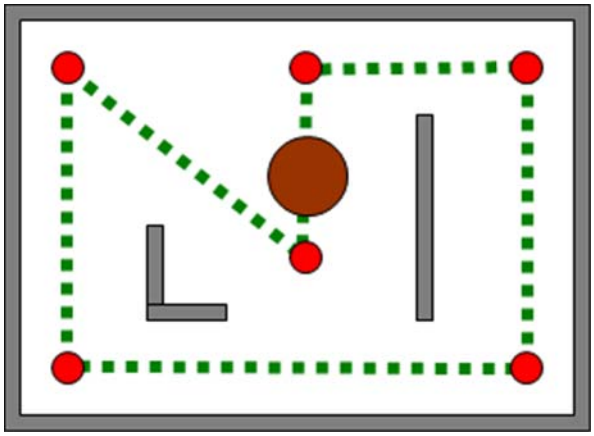
PART 1 – Wall Following:

Set up the environment as shown in the picture here. Write a program called **WallFollow.spin** that will allow the robot to follow along a wall using the front and side IR sensors As the robot follows the wall, it will need to constantly adjust the alignment of the robot accordingly. For your final test run ensure that your robot begins roughly at the bottom right of the environment facing upwards (red dot shown in picture). You may have to adjust your motor constants and speed so as to allow enough time to detect the obstacles properly … but try to get your robot to move reasonably quick, allowing it to slow down and perform any necessary maneuvers in the corners as necessary.



To save yourself some agony, you should make use of the beeper for debugging (e.g., beeping different ways when the alignment is lost or when it is trying to turn in a corner etc.). Using the **RobotTracker**, record a short **.avi** video (called **WallFollow.avi**) of the robot doing one complete wall following cycle. Make sure to also save a screen snapshot showing the actual path that the robot took (called **WallFollow.png**). Make sure that the image is shown in the snapshot so that the environmental setup is clearly discernable.

PART 2 – Point-to-Point Navigation:



Set up the environment as shown in the picture. Write a planner called **NavigationPlanner.java** that attempts to help the robot navigate as accurately as possible between points in the environment using the **RobotTracker** (i.e., not using the encoders). Your code should define 6 points roughly as shown in the picture here. These should be chosen using the user-defined path so that they may be displayed afterwards. Your robot should then be placed at one of these points (your choice) and then should follow the sequence of points as shown in the picture here. The robot should travel in "roughly" straight lines and spin at the points shown to face the next point in the sequence. Note: You will often overshoot your position and your spinning due to the delay in the tracking. Your planner code should always monitor the robot's position and orientation. You may either send the robot its current position and desired location, or you may simply continue monitoring the robot and send it commands such as go forward, spin left and spin right … this decision is up to you. You will need to write a new robot program called **Navigate.spin** to accomplish this. Using the **RobotTracker**, run a test corresponding to the sample points shown in the picture here. Make sure to take a screen snapshot (**Navigate.png**) of the entire run showing the actual path taken as well as the user-defined path overlayed so that a visual comparison can be made. Make sure that the image is shown in the snapshot so that the environmental setup is clearly discernable. Record an .avi movie (**Navigate.avi**) showing the resulting navigation.

Submission:

Create a webpage for this course that contains a simple report on your lab (i.e., a description of what you did and what each part of the assignment was trying to do). Include downloadable links for all of your code as well as any snapshots, trace files and videos. There is no need to make the webpage beautiful … but that's up to you. Login to Carleton's WebCT system and submit a link to your website. Make sure that the website is up and running at least until your code is marked. If you have never created an HTML page before, you can use Mozilla's **Sea Monkey** browser which has a built in editor that allows you to make simple pages. Always keep a backup of all your work (perhaps on a USB key or burn a CD). Here is a summary of what to hand in:

JAVA CODE:
- `NavigationPlanner.java`

SPIN CODE:
- `WallFollow.spin`
- `Navigate.spin`

VIDEO CLIPS:
- `WallFollow.avi`
- `Navigate.avi`

SCREEN SNAPSHOTS:
- `WallFollow.png`
- `Navigate.png`