Homework 4:

Kinematic Feedback Control

Assigned: Thursday, Sept. 25
Due: Thursday, Oct. 2 at the beginning of class (11:10AM)

In this assignment, you will begin using the Player/Stage robotic simulation environment. This homework begins with instructions on how to set up your simulation environment the simulation software.

> **A note on collaboration:** For this assignment, I highly encourage discussions amongst yourselves for the purposes of helping each other learn about Player/Stage, and/or to help each other get everything set up and running. Talk freely, and help each other as needed. Use Piazza to start discussions with the class as a whole. However, each person must individually prepare all the material to be turned in. No sharing of output or textual descriptions (as outlined below) is allowed.

The Player/Stage simulation system is a 2D, physics-based robot simulator. It is open source, and operates under Linux (any of the Hydra or Arc machines in the EECS labs should work) using C, C++, and Python. (It also works with other languages, but not all the languages have the same supporting examples or are as well supported. The examples we provide are in C++.)

Part 1: Getting Player/Stage Running

1. *[Optional, but perhaps helpful]* **Background Reading:** "Most Valuable Player: A Robot Device Server for Distributed Control", by Gerkey et al., is located on the course web site:

   http://web.eecs.utk.edu/~leparker/Courses/CS494-529-fall14/Handouts/Player-Stage-paper.pdf

   This paper provides some background on how the Player/Stage driver and simulator is set up. This is just for your information, in case you are curious. The paper is rather old at this point, but is still considered the original reference for Player/Stage.

   Some online documentation on Player/Stage is available in the public domain, here: http://playerstage.sourceforge.net

   However, keep in mind that this is open source software, created by a crowd of roboticists. So, the documentation is not always complete, and is not always correct. This is the nature of life these days in the open source world. So treat it as a real-world experience, because most open source projects have this nature. You are free to add to the world-wide body of knowledge, through your own experience!

   (Nothing to turn in for this step.)

2. Set up your simulation environment

- Follow the instructions in the Player/Stage Getting Started Guide, located at:

  http://web.eecs.utk.edu/~leparker/Courses/CS494-529-fall14/Homeworks/PlayerStageGettingStarted.html

- At this point, you should be running the robot using simple.cfg and the simple-robot program (as outlined in the Getting Started Guide).

- Explore various menu options on the pull-down menus of the Stage simulation window. For example, note that you can turn on a trace of the robot's motion using a pull-down menu in the stage simulator (called footprints), and that you can create a screen dump of the simulation window using another menu option. You can also turn on and off various sensor displays. Also note that you can drag the robot around using your cursor.

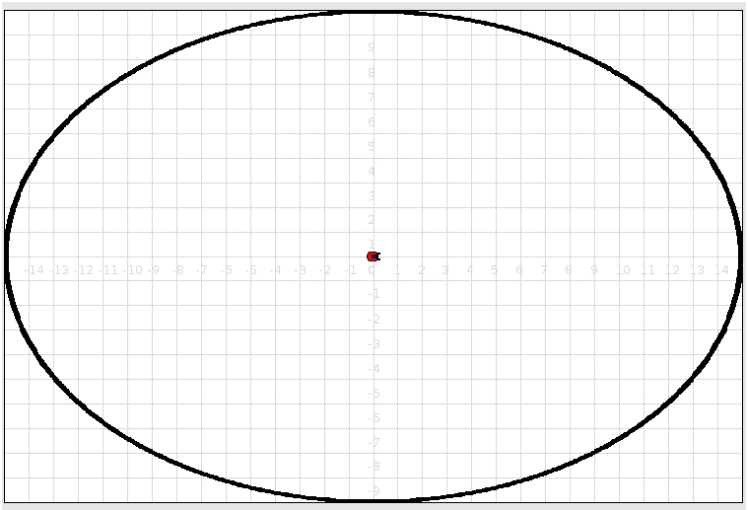
WHAT TO TURN IN FOR PART 1:

- A screen dump of your robot's motion (including the trace of its motion), after running for a few seconds using simple.cfg and simple-robot, displayed in a pdf file. Name this file "<your-last-name>-HW4-Part1.pdf".

Part 2: Implementing Kinematic Feedback Control

In this part of the homework, you will implement the feedback control approach for a differential drive vehicle, as discussed in Section 3.6.2 of your text. You will illustrate this control by driving your robot through a series of goal poses (specified below). Your vehicle should stop at each pose, then move on to the following pose, until the last specified pose is reached.

The sample files provided in Part 1 include the .cfg and .world files you need for this part of your homework (called HW4.cfg and HW4.world). The defined robot is the differential drive Pioneer 2DX robot. This setup uses the "rink" environment and starts the robot at (0, 0, 0). Here is how your starting simulation should appear:



In writing your code, you should begin with the simple-robot.cc sample code provided (in the "examples" directory). The simple-robot.cc code shows you how to connect to the robot, how to get its pose, and how to move the robot. You will not be using any other sensors for this homework other than the position2d proxy (which acts like noise-free GPS).

To test out your kinematic control, move your robot through the following series of goal poses; remember, the robot should stop at each pose:

(0,0,0) → (5,3,0) → (2,7,270) → (-7,4,0) → (-7, -3, 0) → (0, -3, 90) → (7, -5, 180) → (0, 0, 0)

WHAT TO TURN IN FOR PART 2:

(1) Your robot control code, along with your .cfg and .world files, and any other supporting files needed to re-run your code on the Hydra machines. (Files can be named descriptively.)

(2) A README file that says how to run your code.

(3) A screenshot (or set of screenshots) of the robot trace moving through each goal pose noted above, from the starting position to the final goal pose. This/these screenshots should be displayed in a single pdf file. Name this file "<your-last-name>-HW4-Part2.pdf".

Turning in your project

Put all of the required material (Parts 1 and 2 above) into a single tar or zip file (compressed if needed). Then, submit the file using the UTK course Blackboard.