

## CMRoboBits (15-491) Fall 2007

**Homework 1 – Due Wednesday, September 12<sup>th</sup> 2007 at 11:59pm**

<http://www.andrew.cmu.edu/course/15-491/>

### Introduction

In this homework you will model and control the differential drive of a **Scribbler Robot**.

### Submission

Your entire homework should be submitted by copying it into the “**dropbox/lab01**” folder on your personal AFS space (located in /afs/andrew.cmu.edu/course/15/491/students) before the due time.

Please read the website for more information about how to access your AFS dropbox.

Please submit a write-up (.pdf or .ps) containing your answers to the non-code parts such as your graphs and formulas (place it into the same dropbox/lab01 directory).

### Part 1. Calibrate your robot's turning radius (50pts).

The Courseware Lab Tutorial 2 (“Advanced Motion”) makes use of the “TurningRadiusToWheelPowers” service. The TurningRadius input argument is of type double, however it is not entirely clear how this translates to an actual turning radius of your particular physical robot. In this part of the homework you will calibrate a **Scribbler Robot** to allow the conversion of a real-world distance value (in cm) to a “TurningRadius” value of type double.

#### a) Graphing your Turning Radius (20 pts)

Create a 2D scatter-plot of how TurningRadius corresponds to the actual turning radius of your particular robot measured in centimeters. **Important:** Please represent the input argument “TurningRadius” on the **vertical axis**, and your measured radius (in cm) on the **horizontal axis** (while this might seem counter-intuitive at first, it will make sense later on because you are really interested in the inverse of this relationship).

A measuring tape will be provided at the lab. Hint: one possible way to measure the radius is to start the robot perpendicular to a line and have it run a 180 degree turn. Then measure the distance between the starting location and the location where it crossed the line again. This should give you a simple approximation of the turning-diameter (which is twice the radius).

Make sure to have at least 5 data-points in your graph (preferably more), including turning-radii of 0.0, 0.25, 0.5, 1.0, 1.5 (possibly larger, depending how big the turning radius will become)

It is important that you perform this experiment **with an actual robot (not in the simulator)**.

Please include this graph in your write-up. Since the programming component of this question is trivial and fully explained in the first part of the courseware tutorial 2, you are **not** required to submit any code for this particular question.

**b) Provide a simple function approximating this relationship (20 pts)**

Now that you have created your graph, you should be able to find a simple function that you can use to map any particular target radius from real-world coordinates (centimeters) to the “TurningRadius” of type double. Please supply this function  $f(x)$  in your write-up. In this case, the function’s argument  $x$  is the desired radius in cm, and the function’s result should be the “TurningRadius” value. Try to keep your function fairly simple (please don’t overfit it using some crazy function, such as a higher-order polynomial... your data will naturally be noisy, and we are looking for a simple relationship, possibly of a linear, cubic, or logarithmic nature). It is ok to use a curve-fitting tool (e.g. Excel) to do this.

**c) Plot the above function in your existing graph (10 pts)**

To see whether your formula really matches your data, please plot it as a line through your existing graph (that is, generate a lot of points for various possible inputs  $x$ ). Make sure to cover an input range between at least 0cm and 150cm for this plot. Note that for this question, we only want you to plot the function you found in b) onto the existing scatter plot you made in a). You do not need to re-run the robot or re-measure anything.

**Part 2. Variable degree, variable speed, real-world radius turn control (50pts)**

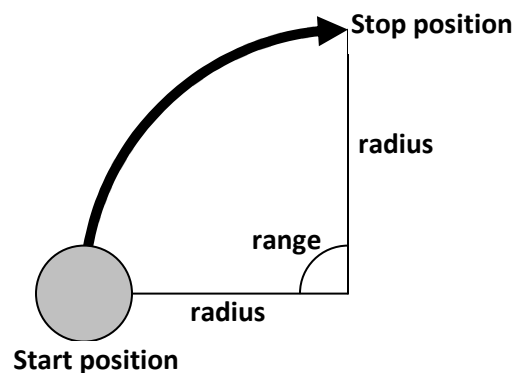
In this part of the homework you will create a VPL program that takes the following three parameters:

**Drive Speed** (type double, range 0.0-1.0): the speed of the robot.

**Turn Radius** (type double, in centimeters): the turning radius of the robot.

**Turn Range** (type double, in degrees): how far to turn.

When starting the robot, your program should ask the user to supply these three values (e.g. using a SimpleDialog). Once all values have been entered, the robot should perform the turn with the properties specified and then come to a complete stop, such as depicted below



Your program should contain and make use of the “TurningRadiusToWheelPowers” service.

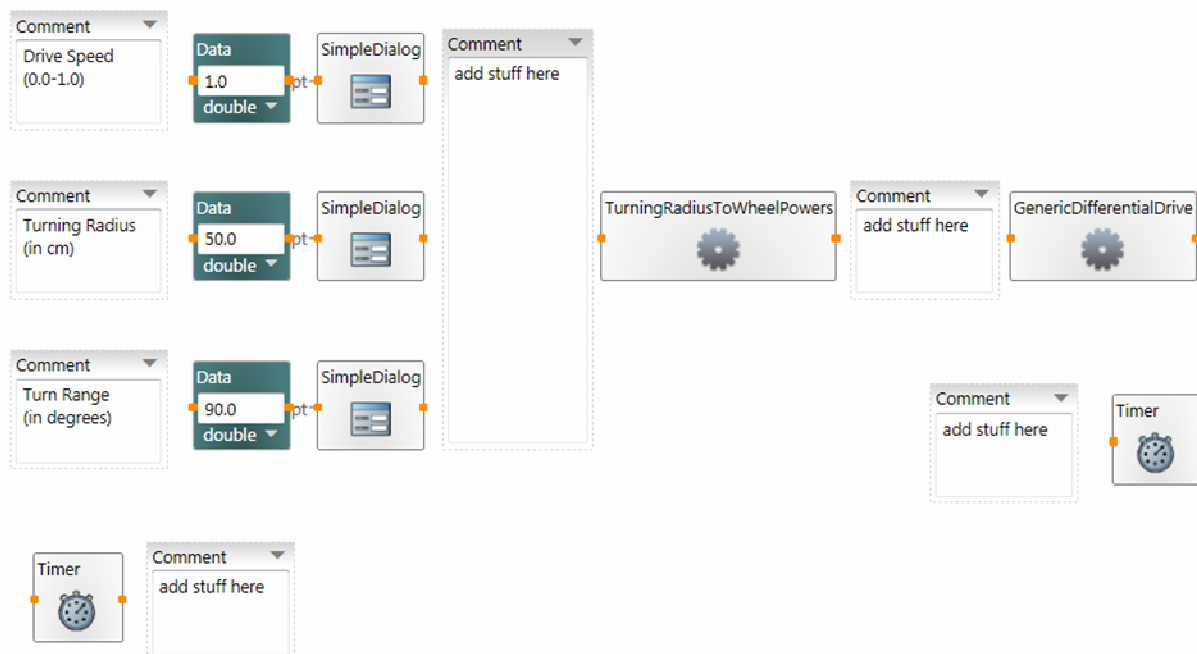
Since we would like the turn radius data to be entered as cm, you will need to use the formula that you came up with in question 1b) to convert this data to a TurnRadius of type double which “TurningRadiusToWheelPowers” will accept.

One limitation of the TurningRadiusToWheelPowers service (as shown in the first section of courseware tutorial two) is that the robot will always drive at its maximum speed. The “Drive Speed” parameter in this question should allow the user to specify any lower speeds as well. So basically it acts as a multiplier to the output of TurningRadiusToWheelPowers. E.g. a “Drive Speed” of 0.5 should drive at half the speed of the default turn speed. You do NOT need to worry about real-world velocities (e.g. cm/s) for this particular variable. It is fine to let it be a simple multiplier in the range of 0 to 1.

The Turn Range is the trickiest part to implement. Since you don’t receive any motor-feedback from the robot, you will need to use a timer to keep track of how long to drive before turning the motors off. You will need to manually calibrate this to figure out how many seconds in time correspond to how many degrees of turn. **Note that this time depends on the speed and the radius!** However, since you do know the speed and the time, you should be able to come up with a very small bit of math to figure out how long to run the motors for (think about computing the distance that the robot is driving...think circumference of a circle....remember that  $\text{velocity} = \text{distance} / \text{time}$ ).

This programming assignment has a couple slightly tricky parts to it. We strongly recommend that you perform the Courseware Lab 2 tutorial first (at least the first half of it which explains how to use the timer). Besides the timer, you will also need to make use of join and merge.

Below is a screenshot of what you should start working with (in particular the three inputs).



**Important:** Please try to comment your VPL diagram by using comment boxes, or the inherent comment property that most services carry.

Also, please add anything you would like us to know about your project to your write-up.

Please submit your entire project (the entire folder, not only the mvpl file) by placing it in your dropbox/lab01/ folder (see submission instructions at the beginning of this assignment).

**Good Luck!**