Lab 1 Getting to Know Your Robot:

Locomotion and Odometry

(Demonstration due in class on Thursday)

(Code and Memo due in Angel drop box by midnight on Thursday)

Read this entire lab procedure and complete Part 1 before coming to lab.

Purpose: The purpose of this lab is to confirm that you have all of the necessary software installed on your computer to connect to and program the Traxster II. The secondary goal is to get the Traxster II moving and to examine problems with raw odometry for pose estimation.

Equipment: Base Robot
Masking Tape
Ruler

Software: Microsoft Visual Studio.Net 2008 with C#
Serializer.NET library and firmware
Bluetooth transmitter

Part 1 – Software Installation

Microsoft Visual Studio with C#

If your laptop did not come with Microsoft Visual Studio, please follow these directions to install it.

1. Click Start -> Control Panel
2. Click "Run Advertised Programs"
3. Select Microsoft Visual Studio 2008 Professional
4. Click Run
5. Click Next through all of the default options

Serializer library

Click the following link to download the Serializer library

http://www.roboticsconnection.com/multimedia/libraries/Installer_15.msi

Bluetooth Transmitter

You will connect to the Traxster II through a Bluetooth Serial Port. If your laptop did not come with a Bluetooth radio, please follow these directions for the HP nw8440. *(If you have a different computer search the web for the driver)*

1. Log into your computer as *localmgr*
1. Download and run the HP Integrated Wireless driver installer (sp38145.zip), in the course Angel folder for Lab 1. This folder contains documentation files for the robot hardware and software in this folder. This folder also contains your weekly lab assignments.
2. When prompted, log off and log back on
3. When you log back on, the Bluetooth icon should appear in the lower right hand corner of the screen. Double-click this icon to run "Initial Bluetooth Configuration Wizard".
4. When you reach "Bluetooth Services" make sure that "Bluetooth Serial" is checked.
5. Complete the configuration wizard and click "cancel" to exit setup when finished.

Part 2 – Download the application code

1. Create a folder for all of your course programs
2. Go to the course Angel folder for Lab 1. Download the *Demo_App.zip* file and save it to the Lab 1 files on your computer.
3. Unzip the folder into your Lab 1 files. The code for this executable is located at Demo_App\IRSonar_App\Demo_app.sln. This program contains code for all of the sensors and peripherals on the robot and will be used to illustrate the Traxster II's capabilities.

Part 3 – Connect to the Traxster II

1. Turn on the robot
2. Confirm that the power light is on for the Bluetooth Serializer V1.0
3. Right click the Bluetooth icon in the tray on the bottom right hand of the screen
4. Click Explore "Explore My Bluetooth Places"
5. Click "Entire Bluetooth Neighborhood"

6. Find eb100. If you are in the classroom full of robots there will be many of them! Each Bluetooth's receiver has a serial number and it should be on a label on the robot.
7. Right click properties on each eb100 until you find the one with your serial number (i.e. Device address: 00:0c:84:3f:e9).
8. On the Authorization tab check Bluetooth Serial Port
9. Click OK
10. Right click Pair on the icon (the pass code is "0000")
11. Double Click on the eb100 icon and it should display "A# serial port on eb100 not connected"
12. Double click the icon and it should now show that you are connected to the robot.
13. Right click on the icon and the General tab will show which COM port the Bluetooth Serial port for the robot is connected on. You will need this number for your code!
14. *(Optional: If you use the Bluetooth Setup Wizard you can configure your device to show the name of your robot, i.e. "Bart" under My Bluetooth Places versus eb100 this will save you time in the future when you connect to your robot.)*
15. My Bluetooth Places should now show "eb100 <RobotName> Connected: COM#"
16. The robot should now be connected and ready to use.

Part 4 – Robot Demo

1. Run the *Demo_App\bin\Debug\Demo_App.exe* application
2. Make sure that the COM PORT on the GUI is the same as the Bluetooth serial port where the robot is connected.
3. After confirmation of connection, experiment with the locomotion, odometry and sensors. (Make sure the robot is clear of tables and legs!!) Verify that the display is consistent with the robot performance, if you find any inconsistencies take note of them and get the technician in the parts room to correct it.

Part 5 – Reviewing the Code

1. Open the solution (*IRSonar_App/Demo_App.sln*) in Microsoft Visual Studio
2. Examine the code and comments in *DemoForm.cs* and try to get some idea of how the program works. The code and comment format for this program is the standard you

should follow for your submitted code. You should also add additional comments to assist you in understanding the program flow in Visual C#.

3. Expand 'References' in the Solution Explorer, and ensure that *"RoboticsConnection.Serializer"* is properly resolved. This means it is listed under References, and that it has a white box icon next to it. If it has a yellow icon, then it isn't resolved. To resolve it, try double clicking on the 'RoboticsConnection.Serializer' namespace. If that doesn't work, then simply right click on "RoboticsConnection.Serializer"' namespace, then 'Remove'. Now, right click on 'References', and then select 'Add Reference'. An Add Reference window pops up. Make sure you're under the *.NET* tab, then scroll down to 'SerializerLib', and select it. It should now be in the References section correctly
4. Build the solution (F7)
5. Run the application (F5)
6. If the following error prints on the Output window after you connect, disconnect from the robot and cycle the power and connect again.

   (A first chance exception of type 'System.TimeoutException' occurred in System.dll)
7. You know the robot is successfully connected when the IR data is changing on the GUI.
8. Confirm that the code works the same as the *Demo_App.exe* application.

****

LAB PROCEDURE

****

Part 1 – Straight Line

Now that we have code to control the robot, let's measure how well odometry performs. As you may recall from Part 1, the robot relies on odometry to determine how far it has traveled, and how far it has turned. If odometry readings are perfect, then the robot movement should be accurate and repeatable.

1. Select 4 different distances (6 in, 12 in, 18 in, 24 in) for the robot to drive and mark the start and stop positions with masking tape on the floor

2. Drive the robot for each distance 5 times generating 20 data points and record the data on a table. Perform an error analysis between each data point and the desired distance. Also, calculate the average distance traveled and state the shortest and longest distances. Also, state the standard deviation and create an x-y scatter plot of the data for inclusion in the lab memo.
3. Move the robot to the hallway and repeat part 2 and discuss the differences in the robot performance on a different surface. Does the odometry error change based upon the distance driven? What about if you change the robot speed, how does this affect the odometry error? Also find the standard deviation of the data and create an x-y scatter plot of the data for inclusion in the lab memo. *(You don't have to take 20 data points just enough to adequately answer the questions).*

Part 2 – Turn Angle

1. Select 4 different angles (90°, 60°, 30°, and 15°) for the robot to turn.
2. Turn the robot for each angle 5 times and repeat the data analysis from Part 1.
3. Does the error increase or decrease with rotation angle? What about if you change the robot speed, how does this affect odometry error?

Part 3 – Square Path

Write a program to move the robot in a square path with sides between 2 and 3 feet (see Figure 1).

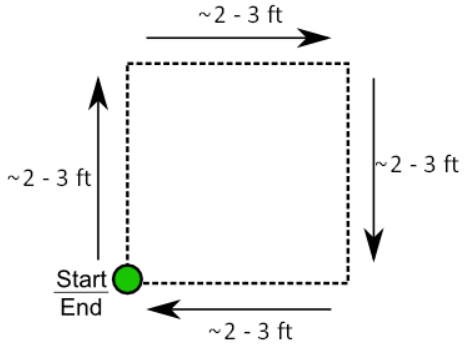

Figure 1: Square Robot Motion

1. Place masking tape on the ground where the robot will start.
2. Run the *"Square"* program and place masking tape at each of the other three corners.

3. Move the robot back to the start point, run the square code, and place a marker where the robot turns and ends.
4. Repeat this 5 times and each time measure the distance between each pair of points (the correct corner and the actual) and compute an average. This is the average error.

Part 4 – More Paths

1. Write a *"Circle"* program to move the robot in a circle with a diameter between 2 and 3 feet (see Figure 2).
2. Write a *"FigureEight"* program to move the robot in a figure eight using two circles (see Figure 3).

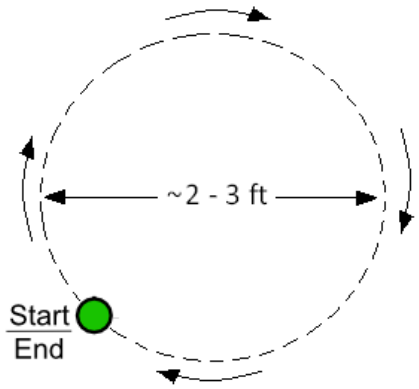

Figure 2: Circle Robot Motion

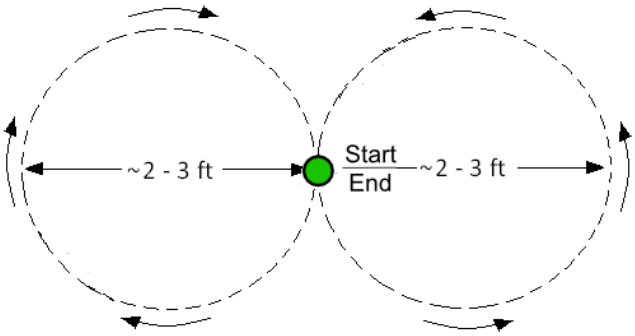

Figure 3: Figure Eight Robot Motion

More questions to answer in the lab memo

1. What are some sources of the odometry error?
2. How could you correct for this error?
3. How could you improve the three motions (*Square, Circle, FigureEight*) programs?

Memo Guidelines:

Please use the following checklist to insure that your memo meets the basic guidelines.

✓ Format
- Begins with Date, To , From, Subject
- Font no larger than 12 point font

- Spacing no larger than double space
- Includes handwritten initials of both partners at the top of the memo next to the names
- Written as a paragraph not bulleted list
- No longer than three pages of text

✓ Writing
- Memo is organized in a logical order
- Writing is direct, concise and to the point
- Written in first person from lab partners
- Correct grammar, no spelling errors

✓ Content
- Starts with a statement of purpose
- Discusses the strategy or pseudocode for implementing the robot paths (may include a flow chart)
- Discusses the tests and methods performed
- States the results including error analysis
- Shows data tables with error analysis and required plots or graphs
- Answers all questions posed in the lab procedure
- Clear statement of conclusions

Grading Rubric:

The lab is worth a total of 30 points and is graded by the following rubric.

Points	Demonstration	Code	Memo
10	Excellent work, the robot performs exactly as required	Properly commented, easy to follow with modular components	Follows all guidelines and answers all questions posed
7.5	Performs most of the functionality with minor failures	Partial comments and/or not modular with objects	Does not answer some questions and/or has spelling, grammatical, content errors
5	Performs some of the functionality but with major failures or parts missing	No comments, not modular, not easy to follow	Multiple grammatical, format, content, spelling errors,

			questions not answered
0	Meets none of the design specifications or not submitted	Not submitted	Not submitted

Submission Requirements:

You must submit properly commented code for the square, circle and figure eight by midnight on Thursday. Your code should be modular with functions and classes in order to make it more readable. You should use buttons, labels, input boxes on the GUI, and/or the keypad to switch between programs, distances or angles. You should use the buzzer, speech module or LCD to indicate changes in state. You must also submit a memo for Lab 1 by midnight on Thursday.