Lab 2 Random Wander, Obstacle Avoidance, Wall Following

(Demonstration due in class on Thursday, 3/26/09)

(Code and Memo due in Angel drop box by midnight on Thursday, 3/26/09)

References: http://roboticsprimer.sourceforge.net/workbook/Sensors:Exercise2:Infra-red_Sensor

http://roboticsprimer.sourceforge.net/workbook/Sensors:Exercise_4:IR_%26_Sonar_wall-following

Purpose: The purpose of this lab is to develop a random wander routine for the Traxster II. The secondary goal is to improve the random wander routine by creating an obstacle avoidance behavior using infrared sensors. The basic behavior of the robot will be to wander in the environment unless it encounters an obstacle. It should then navigate around or away from the obstacle and continue to wander. Finally, integrate wall following which is very similar to obstacle avoidance such that if the robot detects a wall it should then follow it. This is called *subsumption architecture* which we will discuss in more detail later. It is a hierarchy of behaviors where in this case wander is the lowest level, wall following is mid-level and obstacle avoidance is the highest level.

Equipment: Base Robot (4 infrared sensors)

Obstacles, Walls

Software: Microsoft Visual Studio.NET 2008 with C#

Serializer.NET library and firmware

Bluetooth transmitter

Part 1 – Infrared Sensors

There are four infrared sensors on the Traxster II that we will use for obstacle avoidance and wall following. The sensors are connected to the analog I/O on the serializer (0: front, 1: left, 2: right, 3:back). The Sharp GPD120 sensors measure between 1.5" and 12". The following link provides documentation on linearizing the Sharp GPD120 Infrared Sensor Data (Linearizing Sharp Range Data). It is possible to determine a method to linearize the relationship between sensor detection distance and output voltage (see Figure 1). The relationship between voltage and range is $1 / (R + k) = m * V + b$ where R is range and V is voltage. It is possible to linearize

this equation by finding values of m and b. Fortunately, the technical support at Robotics Connection has written code in the Serializer service to calibrate these sensors and we will use the linearized output.

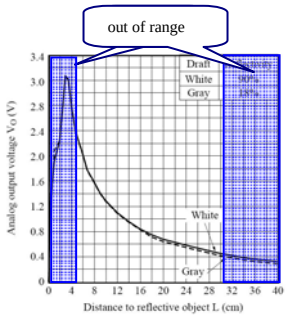
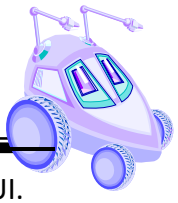


Figure 1 : GPD120 Analog Voltage versus detection distance

1. Download the *Lab2.zip* file from the Lab 2 folder in Angel
2. Open the solution (*Lab2.sln*) in Microsoft Visual Studio
3. Examine the code and comments in *Form1.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.
4. 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
5. Build the solution (F7)
6. Run the application (F5)
7. Turn on the robot
8. Make sure that the COM PORT on the Analog & Digital I/O GUI is the same as the Bluetooth serial port where the robot is connected.
9. 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)
```



10. You know the robot is successfully connected when the IR data is changing on the GUI.
11. After confirmation of connection, experiment with the Analog & Digital I/O GUI and make sure you understand how to interpret data from the IR sensors and use the feedback to send an output to the robot buzzer or motors.

Part 2 – Random Wander

Create a random wander routine that the robot can use to explore the room. This can be done by generating a random number that represents a factor to scale drive time, turn angle or drive distance and then move the robot. (Watch for table and human legs!)

Part 3 – Obstacle Avoidance

Now that you are familiar with digital I/O, analog I/O and how to move the robot, create an obstacle avoidance behavior. The robot should drive forward until an obstacle is encountered. The robot should drive away from the obstacle and continue to go forward. The most basic behavior would be to turn away based upon proximity as determined by one of the infrared sensors. Use the buzzer on the robot to indicate when an obstacle has been encountered. The LEDs on the robot can be used to indicate whether the robot is in drive or avoidance behavior. Your program should provide a method to get the robot 'unstuck' if it approaches any local minima points (i.e. oscillates between two obstacles). Your program should be as modular as possible with multiple subroutines and behaviors that will be integrated in subsequent programs. Devise a method to test and confirm that your program works correctly and present the results in the laboratory memo (see Figure 1).

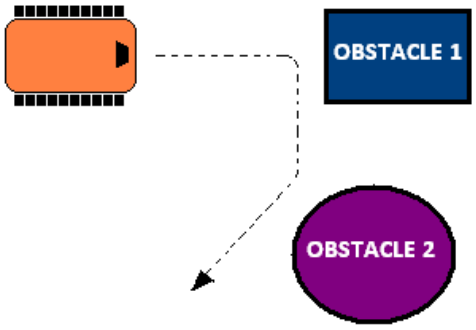


Figure 1: Obstacle Avoidance Example

Part 4 – Improved Random Wander

Now improve the random wander routine by integrating obstacle avoidance. The robot should wander randomly until an obstacle is encountered. The robot should drive away from the obstacle and continue to wander. Use the buzzer on the robot to indicate when an obstacle has been encountered. The LEDs on the robot can be used to indicate whether the robot is in wander or avoidance behavior. Your program should provide a method to get the robot 'unstuck' if it approaches any local minima points (i.e. oscillates between two obstacles). Your program should be as modular as possible with multiple subroutines and behaviors that will be integrated in subsequent programs. Devise a method to test and confirm that your program works correctly and present the results in the laboratory memo.

Part 5 – Wall Following

Now modify the obstacle avoidance program so that the robot uses the IR sensor to detect a wall on its left or right side and attempts to follow it while maintaining a 5 inch distance. You should attempt to address issues such as doors, getting unstuck from corners and turning corners (see Figure 2).

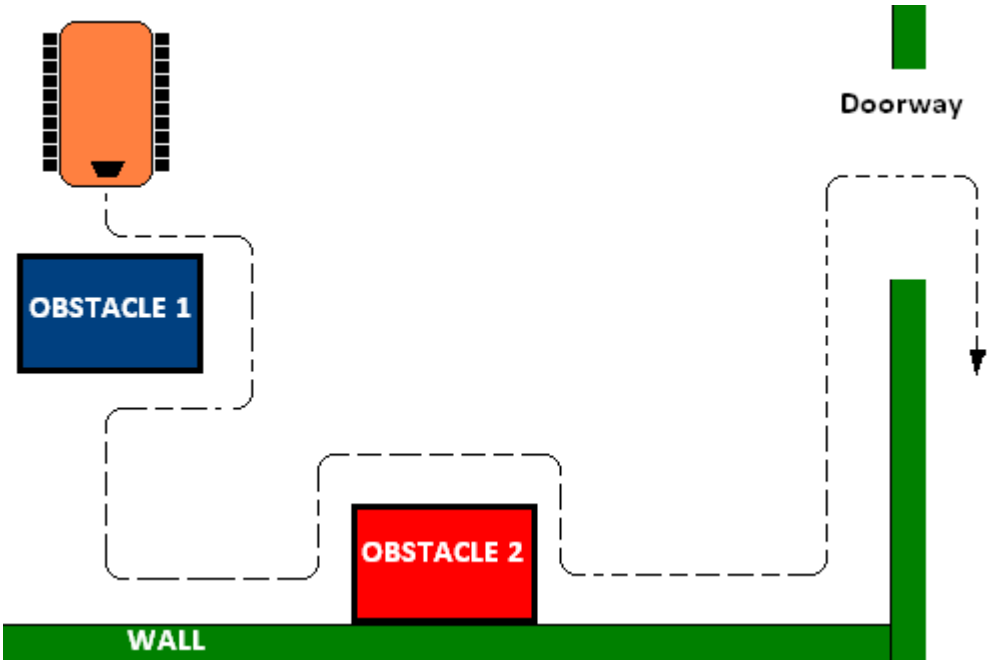


Figure 2: Wall Following Example

Questions to Answer

1. What was the general plan you used to implement the random wander and obstacle avoidance behaviors?
2. How could you create a smart wander routine to entirely cover a room?
3. What kind of errors did you encounter with the obstacle avoidance behavior?
4. How could you improve the obstacle avoidance behavior?
5. Were there any obstacles that the robot could not detect?
6. Were there any situations when the infrared sensors did not give you reliable data?
7. How did you keep track of the robot's states in the program?
8. How did you get the robot unstuck from corners?
9. What do you do when the robot encounters an obstacle next to the wall?
10. How does the robot handle doorways in the wall?

Submission Requirements:

You must submit properly commented code for the square, circle and figure eight by midnight on Thursday, 3/26/09. You must also submit a memo for Lab 1 by midnight on Thursday, 3/26/09. Please insure that your memo meets the basic guidelines.