Driving, Path Finding, and Obstacle Avoidance

## Purpose

The purpose of this part of the project was getting the readings from the sensors needed to obtain useful information from the robot bases, as well as getting basic functionality for driving, route finding, and obstacle avoidance.

## Getting Started

After installing all the necessary software, I started this project by using the “testaria.cpp” file to test all of the basic functions the robot is capable of. This also helped me get an understanding of how to initialize the communication port between the robot and the PC board used to control it. After that I started to build on basic functions of this base until they became useful.

## Robot Base

The MRPT libraries contain a class called CActivMediaRobotBase for initializing the Robot Base. A CActiveMediaRobotBase object contains information from the sensors, bumpers, battery, as well as the motor settings and feedback. I started this project by first figuring out how the sonars work. Each sonar has an individual ID and will return a distance in meters upon request. The IDs for the sonars can be seen in the image below:



A “robot” object for the CActivMediaRobotBase can be initialized with:



The sonars can be checked with a call to the following functions:



Other data can be retrieved from the robot in a similar manner including battery, current velocity, bumper status, as well as other information.

## Robot Movement

To get the robot to move and turn the CActivMediaRobotBase object must also be used. The method of CActivRobotBase that does this is *robot.setVelocities(double linear\_velocity,double angular\_velocity),* where **linear\_velocity** is the velocity in meters per second (positive for forward or negative for backwards) and **angular\_velocity** is in radians per second. To make the base easier to control I created a move forward and turn function. These functions are *moveForward(double distance)* and *turn(double angle),* where **distance** is in meters and **angle** is in radians to turn. M\_PI is useful in the turn function. These functions are used in traveling from one node to another.

## Route Finding

One of the goals of this prototype was to have it be able to navigate the lower floor of the Fourth Avenue Building. To do this a floor plan for the FAB was obtained from <http://www.fap.pdx.edu/floorplans/detail.php?buildingID=12>. To get the directions from one location to another they were measured on the map and slightly tweaked after a check. For route finding a graph structure is used to represent the map of the FAB. Each location in the graph is a node that contains the name of the location as well as the index. Below is the current map:

Edges must then be added to the nodes that are directly connected to each other. As of now a method must be created that gives instructions from one location to another as well as instructions how to return back from that node. Here is an example how the robot can get to the Men’s Room from the Robotics Lab as well as the return route. Note at each node the robot should have a defined direction that it faces.



If the robot cannot get to the destination from traveling any of its edges directly, the graph class will return a stack with the nodes it must travel. It uses depth first search for returning this stack. It will then pop off all the locations into a location array and go through a loop to traverse the locations in the array one at a time. As of right now, there is likely to be error and no way for the robot to know that it is off. Our team was having trouble integrating the kinect to this project, but this should be one of the concentrations of the next teams working on this project. The idea was to have images stored for each location and when the robot reaches its destination, or at intervals/checkpoints, along the way the robot would correct itself. It would be very interesting to see how genetic algorithms could optimize this calibration.

## Obstacle Avoidance

A crucial function of the robot is to avoid any collisions along its path. To do this the robot must continuously check its sonars as well as its bumper information. A temporary function is in place to avoid collisions, but a better method should be developed. The diagram below shows how the robot tries to avoid an obstacle when it is sensed half a meter from its front. It only uses the front two so this right now is not an efficient way of doing it.

.5 m

## Created Functions

**moveForward(double distance) –** This function was created for traveling forward. A distance in meters needs to be put in the parameters to move the robot forward. Delays may need small adjustment.

**Turn(double degreesRadian) -** This function was created for turning left and right. To turn right a negative angle in radians should be entered, to turn left a positive value should be entered. M\_PI is useful for this turn distance (M\_PI/2 = 90 degrees).

**getSonarDist(int ID) –** A call to this function will return a distance in meters from the ID put in the parameter. See the sonar diagram for IDs.

**avoidObstacle(double speed) -** This function is the routine to handle the case when the robot senses that an obstacle is a set distance ahead of it. It is checked if this needs to be called every 100ms in the moveForward function. **WARNING: THIS SHOULD NOT BE TESTED ON HUMANS!!!!**

**travelToLocation(stack<location> path, string destination, string currentLocation) -**  This function is passed a stack with a route to follow to the next location in it, as well as the start and end locations. The strings for these locations are the names of the node created upon initialization.

## Future Route Finding

Although basic path finding abilities are now possible with the guidebot, there is still much work to do to get this project at its desired state. The graph needs to be expanded, a calibrate function that implements vision should be created, and the obstacle avoidance method should be optimized to improve performance. The tools that are available to complete this robot are vast and when fully utilized they can eventually serve as a fully functional prototype of a real guide robot as well as a great learning platform for students.