Laser Listener in ROS

To make your robot responsive to Lidar data, you will want to subscribe to the topic on which laser data is published. See www.ros.org/doc/api/sensor_msgs/html/msg/LaserScan.html for a description of the scan message from a laser range finder.

If you run **roslaunch cwru_sim_stage harlie_amcl_2ndfloor.launch** you will find that the cwru robot simulator publishes to the topic "base_scan." (After starting the simulator, type: **rostopic list** to see the topics being published). You can see that this topic is of type "sensor_msgs/LaserScan" (by invoking: **rostopic type /base_scan**). Typing: **rosmsg show sensor_msgs/LaserScan** will display the data structure of the message type LaserScan. The message includes fields to specify the start angle of the scan, the end angle of the scan, the angle increment, the max range, and a list of ranges. For our laser scanner simulation, we have a max range of 80m, the scan starts from -90deg (along the –y axis) and samples at 1-degree steps up to +90deg (along the +y axis), resulting in 181 samples (stored in the "ranges" variable).

You can see some of the data being published by typing: **rostopic echo /base_scan**. This will verify the angle ranges, angle increment and max range.

Subscribing to the base_scan topic is virtually identical to the pose_listener example. An example laser_listener CPP file follows:

```
#include <ros/ros.h>
#include <cwru_base/Pose.h>
#include <sensor_msgs/LaserScan.h>
#include <iostream>

using namespace std;

// For using the LaserScan, see
//http://www.ros.org/doc/api/sensor_msgs/html/msg/LaserScan.html

void laserCallback(const sensor_msgs::LaserScan::ConstPtr& laserScan)
{
    for(uint i = 0; i < laserScan->ranges.size(); i++)
    {
      if (laserScan->ranges[i]<0.75)
        cout<<"ouch, ";
    }
    cout<<endl;
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "laser_listener");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("base_scan",1,laserCallback);
    ros::spin();
    return 0;
}
```

This example may be compiled with **rosmake** (after adding it to your CMakeLists.txt file), and run with the **rosrun** command. If you have the cwru simulator running, you can drag "Harlie" near a wall, or drag one of the orange, square obstacles near the front of Harlie, and the laser_listener node will print out "ouch" if any one of the laser pings has a length less than 0.75 meters.

This illustrates how to subscribe to the base_scan topic and how to get access to laser rangefinder data in simulation. The same technique applies to the actual robot (but confirm the topic name for the actual Sick Lidar).