# Lab 3: Path Service for Turtlebot

Sasha Belotserkovskaya, avb27@case.edu; Kristina Collins, kvc2@case.edu; Shawn Qian, xxq82@case.edu; Connor Wolfe, cbw36@case.edu

## I. Introduction

This assignment presents an example of a programmed path on the Turtlebot platform. It serves to demonstrate ROS service communication for a multipoint path.

## II. Testing

### A. Demonstration Steps

First, connect to Turtlebot using the instructions in the lab assignment. The Turtlebot acts as the ROS master. Recommended to use Google's DNS for stability. Also, note that connecting to the Turtlebot in one terminal does NOT connect you to it in other terminals. To wit, for deeplearning02:

- **host deeplearning02w 8.8.8.8** Take note of the robot's IP, using Google's DNS.
- **ifconfig** Take note of the computer's IP.
- **ping deeplearning02w 8.8.8.8** Check to ensure that data can be exchanged.
- **source /opt/ros/indigo/setup.bash**
- **export ROS_MASTER_URI = http://deeplearning02w.eecs.cwru.edu:11311** Set the Turtlebot as ROS master.
- **export ROS_IP=XXX.XX.XXX.XX** Use the computer's IP here.
- **source /devel/setup.bash** Switch the source to match the location of the written nodes.s

It's wise to verify connection by checking rostopics with the command **rostopic list**. Next, run the program:

- **roscd**
- **rosrun bacon1 bacon_lidar &**
- **rosrun bacon1 bacon_path_service &**
- **rosrun bacon1 bacon_path_client &**

The Turtlebot should start to move through a programmed path. The position number, travel distance and yaw will be printed in the terminal.

### B. Tips and Notes

It's wise to keep an eye on the Turtlebot and make sure it doesn't bump into anything. The instructions above and in the README recommend using the Google DNS and addressing the turtlebots by IP rather than hostname. We demonstrated our code on deeplearning02. Finally, note that connecting to the Turtlebot in one terminal does *not* create a connection in other terminal windows. For this reason, it's wise to use bash commands (CTRL+Z, B+G, F+G) to run both nodes simultaneously in the same terminal window. (This makes processes a little more difficult to identify and kill when necessary, but still saves time).

## III. Testing & Results

We successfully ran the code on deeplearning02, as shown in the video. In STDR, the robot's estimate of its current position is more reliable (although there is still some uncertainty); odometry feedback would help to correct this. For the physical Turtlebot, we reduced the speed to keep it from bumping into things. As with PS2, it was necessary to publish to cmd_vel instead of robot0/cmd_vel.

Our code may be accessed on Github from the following link:

https://github.com/shacks2017/teambacon

## IV. Conclusion

This exercise, like PS1, is an open-loop controller. The ROS service communication mode doesn't provide a method for the client to interrupt the server in the event of a sensor feedback alarm; for that, it's necessary to use an action server.