|  |  |
| --- | --- |
| To: | Dr. Berry |
| From: | Christopher Collinsworth, Jordan Patterson |
| Date: | 12/13/2015 |
| Re: | Lab 2 – Locomotion and Odometry |

The purpose of this lab was to program the robot to move to a given angle and a given goal position, as well as move in square, circular, and figure-8 paths. The UMBARK test also had to be applied to identify Type A and Type B Odometry errors, which were compensated for by calculating a tuning factor.

To implement the Go-To-Angle behavior, we first had the robot spin in the desired direction by running one of the motors at a specified speed and the other motor at the same speed but in the opposite direction. The delay time used for delaying the robot’s movement and reaching the desired angle was calculated according to previous trials of spinning the robot based on a given delay time.

To implement the Go-to-Goal behavior we first had the robot spin to the desired angle, which was calculated using the specified x- and y- coordinates, via the GoToAngle function. We then delayed the robot’s movement for 1 second to allow the motors to stop running. Using the GoDistance function, we calculated the distance for the robot to move by using the x- and y- coordinates, calculated the time for the robot to move, and then moved the robot forward by running both motors at a set speed over the calculated time.

Having the robot move in a square path was achieved by calling the GoToGoal function four times.

Having the robot move in a circular path was accomplished by running the motors at different speeds in the same direction. The delay time for delaying the robot’s movement was calculated by trial and error. The delay time for the counter-clockwise direction was 1.5 seconds longer than the clockwise direction. Having the robot move in a figure-8 path was accomplished by calling the MakeCircle function twice, once in the counter-clockwise direction and once in the clockwise direction. Both motors were stopped in-between the two circles.

3.) The move distance was calculated from the given x- and y- coordinates by using the equation below:

7.) There are a multitude of sources related to the Odometry error. Systematic errors include unequal wheel diameters and misalignment of wheels. Nonsystematic errors include travelling over uneven floors and wheel slippage.

8.) We used the UMBARK method to correct for systematic errors such as those mentioned above by finding adjustment factors to account for repeatable Odometry error. For nonsystematic errors, there was nothing that we could really do besides manually fine-tune our some of our calculations through trial and error.

Programming the 6 keys to move the robot was accomplished by first determining the values of the IR remote control keys that we decided to use. These values were obtained by running the R08\_Remote\_Control program that was built in and pressing the intended keys to display the values on the LCD screen. The 6 keys that we decided to use and the robot movements associated with each key are listed below:

Up arrow – move forward

Down arrow – move backward

Left arrow – turn left

Right arrow – turn right

1 – spin left

3 – spin right

Below is a figure of the remote control:

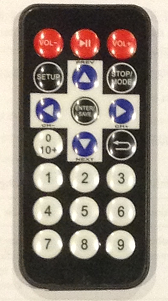


Figure 1. Robot remote control.

We then implemented the key values into our program by associating them with their corresponding movements via switch statements. So depending on which button was pressed, our program would read the key value, go to the specific switch case, and run the code under that case to initiate the desired movement of the robot.

Moving the robot in a circular path was successfully accomplished by repeatedly pressing the right arrow key, causing the robot to turn to the right until completing a full circle. Moving the robot in a square path was not as successful, though. We attempted to move the robot forward via the up arrow key and spin right via the 3 key. This method, however, did not result in a perfect square path because the robot would not spin at an exact angle of 90 degrees. This was due to friction between the floor and wheels and the alignment of the wheels not being perfectly calibrated. Navigating the robot through the obstacle course was successfully accomplished by using the up arrow, left arrow, right arrow, 1 and 3 keys to maneuver the robot through the cones in a zigzagging manner.

In conclusion, the robot performed well and was able to complete all of the specified tasks. The only exception was moving it in a square path. This problem can be resolved by correcting the speed of the robot’s wheels and the time delay to account for the error that resulted from friction between the wheels and floor and the misalignment of the wheels.