Isabelle Brandes
John Wilson
Team R2

## Questions

1. What happens (in terms of the robot's behavior) during the robot.step(TIME_STEP) statement?
    - In the case of the lab, each iteration of robot.step(TIME_STEP) is sending the data in the controller to the robot's sensors and actuators. TIME_STEP represents how long the actions in the while loop will be simulated before continuing upon its next iteration. It detects where the line is in relation to itself, decides whether to turn or go straight, calculates its odometry values, and then sends the velocities calculated to the motors.

2. What happens if your robot's time step is not exactly TIME_STEP long, but slightly varies?
    - If TIME_STEP varies between each iteration, there will be no consistency in its actions. As robot.step is not being called at regular intervals, the robot will act irregularly, as it is being sent instructions to execute for non-consistent amounts of time, resulting in variables that differ from those that we would receive in the real world.

3. What is the ePuck's average speed (in m/s) from Part 1?
    - .125

4. In an ideal world, what should the ePuck's pose show each time it crosses the starting line?
    - (0,0,0) as it looped back to its starting point. But due to slippage and error, this is not the case.

5. How did you implement loop closure in your controller?
    - We used a counter and determined that if passing the starting line, the counter would increment 20+ times, and sharp corners would only increment it once. So our condition is if the counter is above 10, we have reached the start line and we reset to (0,0,0)

6. Roughly how much time did you spend programming this lab?
    - Probably longer than we should have, around 10 hours.

7. Does your implementation work as expected? If not, what problems do you encounter?
    - For the most part, yes. There are some discrepancies in the z-direction, but overall our odometry equations are correct and our controller works well enough.