CSCI 3302 Lab 2

Jack Blackburn Kyle Paris Ben Burkhalter

2. What happens (in terms of the robot's behavior) during the robot.step(TIME_STEP) statemen

The robot.step function is used to synchronize the sensors, motors, and controllers on the robot. Each time it is called, it runs the physics simulation for the duration of time specified in order to update the robots position, sensor values, etc. In our case, we are reading the ground sensor values and updating the wheel velocities.

3. What happens if your robot's time step is not exactly TIME_STEP long, but slightly varies?

Our calculations of pose will be error prone and get worse and worse with each iteration. We depend on the amount of time passing with each step to integrate and calculate the pose, thus if it is slightly off it will throw off our final calculations.

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

.1144675 m/s

5. In an ideal world, what should the ePuck's pose show each time it crosses the starting line?

(0,0,0)

6. How did you implement loop closure in your controller?

We created a state that was called "Done" meaning that the robot was back at the start, determined via the sensor data. If all three of the sensors showed a black reading and the prior state was straight, that was our criteria for when the robot was back at the starting line. When it is here, it stops for a few loop iterations, accelerates, and then resets the odometry and continues with the next loop.

7. Roughly how much time did you spend programming this lab?

3-4 hours designing the logic and implementing the code and then debugging.

8. Does your implementation work as expected? If not, what problems do you encounter?

Our implementation went pretty smoothly overall, we had a bit of trouble keeping up with units while calculating pose but quickly overcame the issue. Also, when handling the loop closure it took a bit of messing around with our code logic to handle this aspect.

We also had trouble after our robot had looped around the lap about 16 times. At this point it turned in a circle forever around the starting line. To fix this we realized that our threshold for sensor data was far too low, at 500, so we raised it up to 800 and this solved the problem.