Project Proposal - Team Blue

Members:

Darrien Lee: dale3920@colorado.edu
Daniel Kim: daki4697@colorado.edu
Xiaodong(David) Zheng: xizh9978@colorado.edu
Ruijiang Ma: chma2866@colorado.edu

Possible Ideas:

1. Maze mapping and navigation (Something like this: https://youtu.be/76blllun09Q)
2. Use lidar sensors to first generate an internal representation of the maze, then apply the uniform search algorithm or A star algorithm to find the fastest route from the starting point to the goal point.

Abstract:

Our project will be based on maze mapping and navigation, which basically consisted of two major components: the first part is maze mapping, in which we let the robot explore the maze and map out a general representation of the maze in 2d; the second part is navigation, where we reset the robot to the beginning state as well as setting a goal state. The robot would, using the mapping from the last step, navigate and plot out a fastest path to the goal state.

Equipment:

1. Epuck robot
2. A maze map created with solid walls

Deliverables: webots world (.wbt file), controller file (.py file)

a. Create a maze world, e puck, and target in the system (walls for the edges of maze)
   Lead: __all__ Deadline: __11/18__
b. Create a mapping system controller to navigate the maze and generate an internal representation.
   Lead: __undecided__ Deadline: __undecided__
c. Implement navigation controller (A* or UCS or Greedy algorithms) to find the best route.
   Lead: __undecided__ Deadline: __undecided__

Implementation Plan

1. Epuck will map out the maze first in its own state controller (“maze_mapping”) onto some sort of world representation (2D array, tuple, vectors, matrix, numpy, etc).
2. Then the controller will switch states to navigation (“maze_navigation”) and find best path towards a target
3. The target can be moved around in real time and the robot will constantly be updating to find the best path towards the target in the maze.

Demo

We are going to have a 2 meter by 2 meter maze with multiple destinations in the maze. We will have the epuck placed at random within the maze. To begin, the epuck will go explore the maze and start mapping the maze using its sensors. After the epuck finishes mapping the maze, it would output the mapping generated somehow(maybe output as a new file). For the navigation phase, we will somehow load the mapping we just got and place a target destination at one of the multiple destinations. Using our navigation algorithm, the epuck will find the shortest route to get to the target destination.