Jay Jeyaruban

Robotics individual report

For this task, we set about assigning sections to work on individually and then pooling our efforts together later for the final result. The initial setup involved Jas working on modifying our wall-follower from a previous task in order to explore the maze. I was working on finding a suitable data structure that we could use to keep track of various factors such as visited points in the maze or the directions there are possible paths. This would then be used to help develop a pathfinding method for the final part of the task.

The first thing I did was research algorithms to use for the task. I came across the FloodFill algorithm and found it easily applicable to what Jas was already working on and it helped provide a simple method for finding the path. The method conceptually revolved around assuming the maze had no internal walls and for each point in the maze, you associate a distance to the goal (in number of movements into adjacent squares). In our 4x4 maze, this meant the top right had a distance value of 0 and the bottom right had a value of 6. As the robot navigates and ‘maps’ out the maze, the distance value at each square in the maze needs to be updated to the minimum of the adjacent squares (where there is a possible path) +1, if the value needs to be changed to match this, the open adjacent squares need to be added to a stack and the distance values in these square needs to be re-evaluated as well additional.

I initially implemented this in the form of a 4x4 2D array consisting of a node structs which each held a bool array of size 4, a bool for visited and an int for distance to the target square. The bool array indicated if the square had a possible path in each of the cardinal directions, true indicates if there is a wall and false indicates there is a possible path. The visited bool is used to indicate if that point has been visited previously which is mainly used in the exploration phase. The final distance int is used in phase 2 of the algorithm. A path can be found to the destination point by going to each square, finding the neighbour, where there is possible path, which has the shortest distance to the target square and then moving towards it. This is done until the target square is reached.

The task of implementing this final part of the algorithm was given to Laura who was added to our group recently.