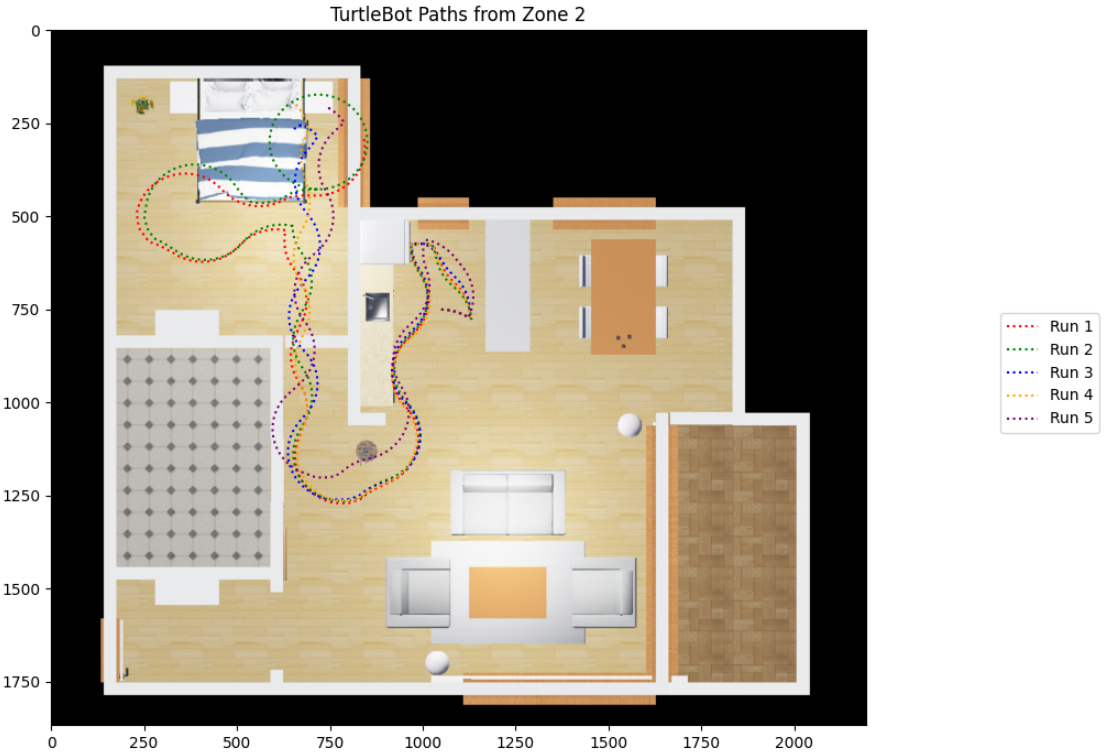
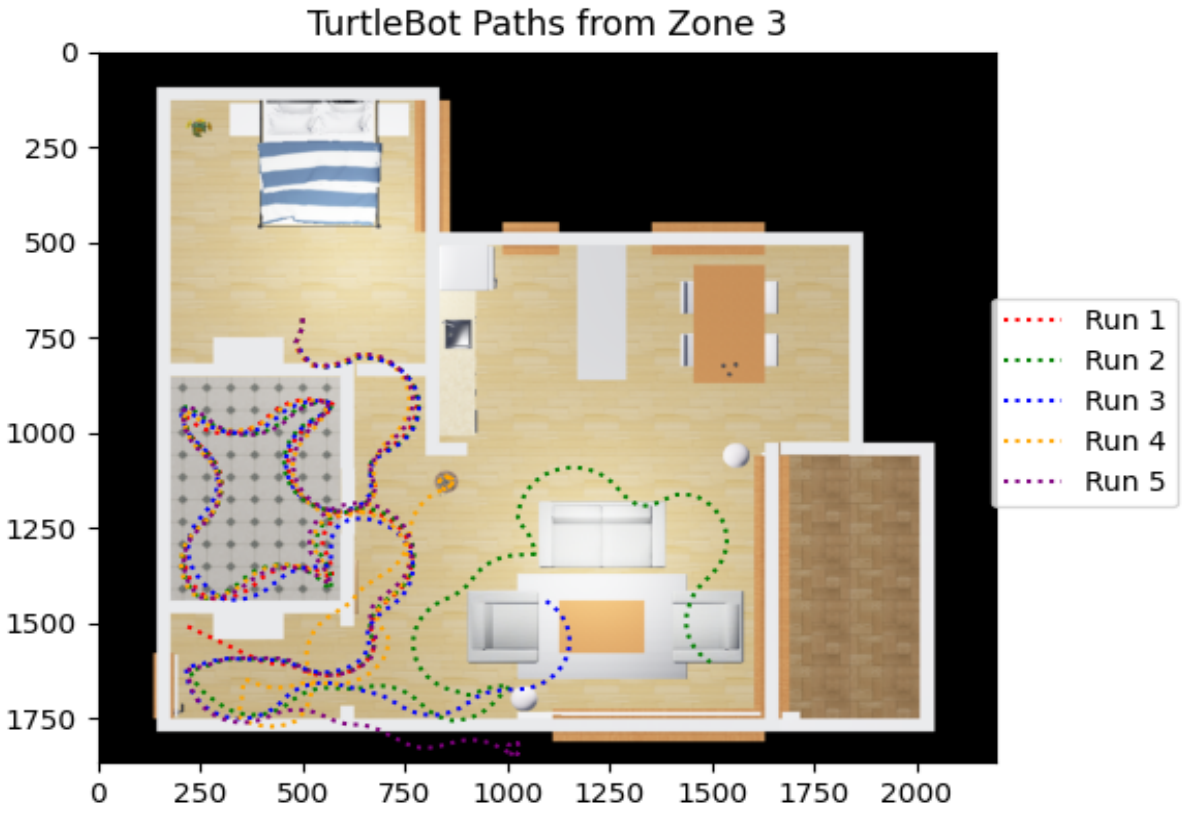
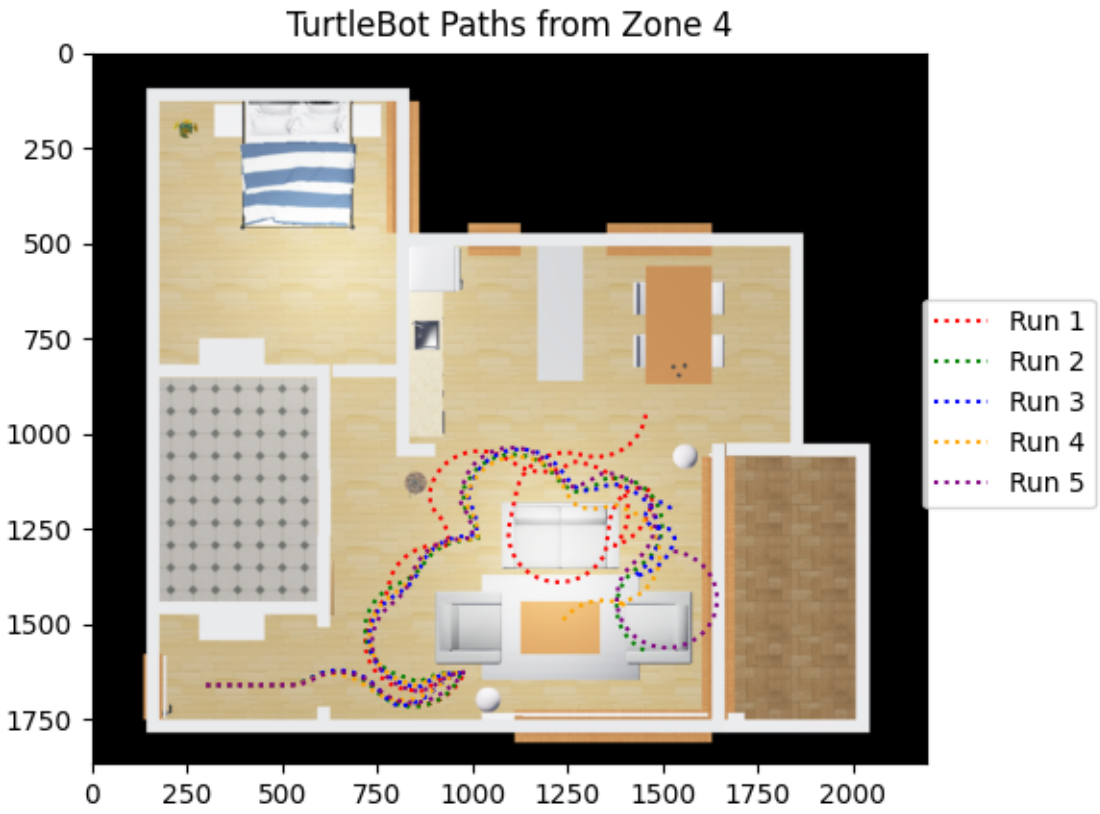
ZONE	TRIAL	TOTAL DISTANCE (m)	FURTHEST POINT (m)
Zone 1	Trial 1	4.00	0.67
	Trial 2	2.11	0.77
	Trial 3	2.53	0.66
	Trial 4	2.78	0.81
	Trial 5	2.57	0.77
Zone 2	Trial 1	19.53	4.92
	Trial 2	23.15	4.83
	Trial 3	14.48	3.65
	Trial 4	14.15	4.00
	Trial 5	13.87	3.62
Zone 3	Trial 1	21.93	5.54
	Trial 2	38.66	7.78
	Trial 3	28.41	6.55
	Trial 4	28.03	6.33
	Trial 5	28.08	7.39
Zone 4	Trial 1	21.61	7.84
	Trial 2	15.59	7.31
	Trial 3	15.89	7.47
	Trial 4	15.10	7.11
	Trial 5	18.02	7.77


TurtleBot Paths from Zone 1
0
250
500
750
1000
1250
1500
1750
Run 1
Run 2
Run 3
Run 4
Run 5


TurtleBot Paths from Zone 2
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000
Run 1
Run 2
Run 3
Run 4
Run 5


TurtleBot Paths from Zone 3
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000
Run 1
Run 2
Run 3
Run 4
Run 5


TurtleBot Paths from Zone 4
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000
Run 1
Run 2
Run 3
Run 4
Run 5

The approach I took to get the Turtlebot as far from its starting position as I could is the wall following heuristic. To do this, I mainly changed the timer_callback function. The main two distances that are needed for this are the minimum distance in the front of the sensor and to the right because I want to follow on the right. I also want the Turtlebot to go through doorways consistently. To go through doorways, I track the most recent minimum LIDAR value on the right. I then subtract the old value from the current value to look for a significant change. This will indicate when it is passing a doorway. When this is detected, the Turtlebot will slow its linear velocity and turn clockwise towards the doorway. If a doorway is not detected, the next check is whether or not there is something in front of the Turtlebot. If there is, linear velocity is zeroed out and turns counterclockwise. The next two conditions check if the Turtlebot is within an acceptable range of distance from the wall on its right. If it is not, it corrects by either turning towards or away from depending on if it is too close or far from the wall. If none of these criteria are met, the linear velocity is set and the angular velocity is zeroed out.

An advantage of my approach is the speed with which the TurtleBot was able to get away from its starting position. It was also simple to implement. I went with implementing this because I wanted to boil the problem down to be as simple as I could make it to still get good results. The major disadvantage is that the Turtlebot doesn’t know if it is stalled. This usually didn’t happen, but it would sometimes on different items around the rooms like the cabinet. This was the issue that my robot had in Zone 1. It would always find the wall in front of it, and it would then get stuck on the lamp at the end of that wall. I couldn’t find a good fix for this within my approach, but it was what I was last attempting to fix.