CS 460 – Assignment 1 Report

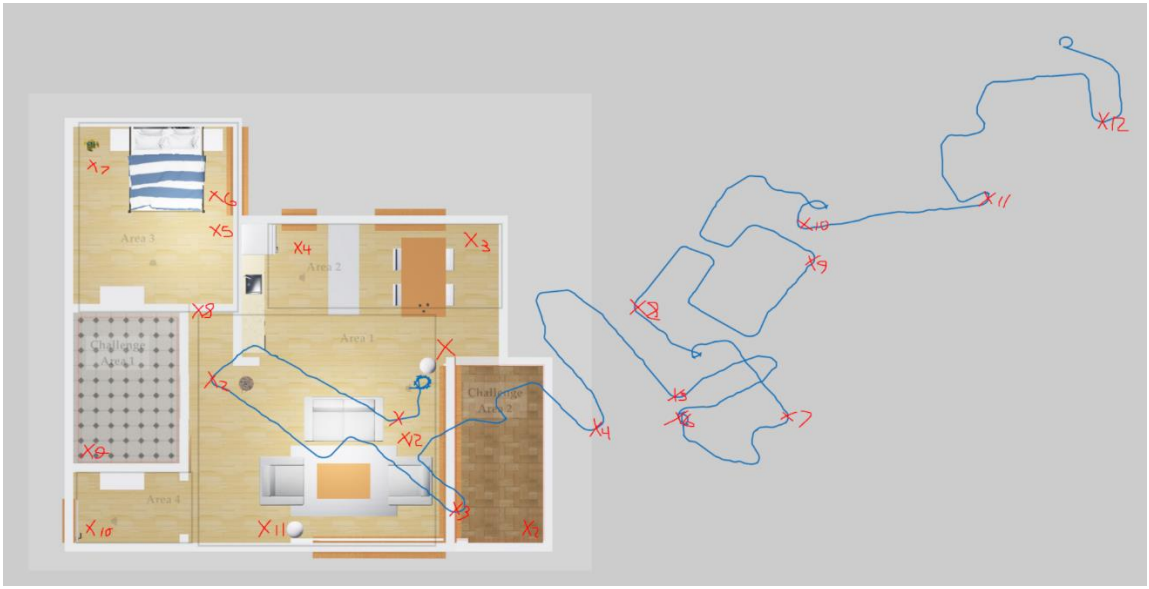
Lilly Eide – rweide@crimson.ua.edu

Sept. 25th, 2024

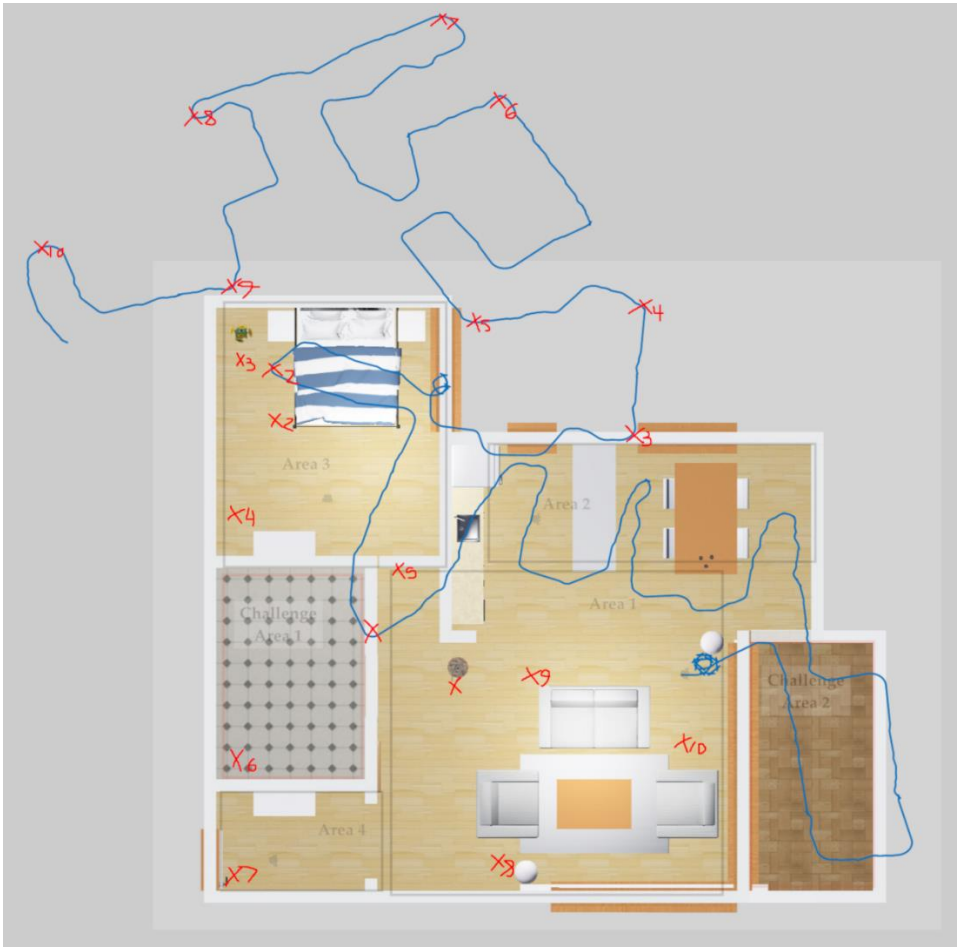
The path lengths were generated using a MATLAB function that I wrote to loop over the logged points, calculating the distance between each point, and summing all recorded distances. Note that due to time constraints, only a single farthest distance was able to be recorded for areas 2, 3 and 4, and only 4 out of 5 distances for the first area. All trials were able to complete a full loop around the entire apartment layout, however.

Area & Trial	**Total Path Length**	**Farthest distance from start**
Area 1, Trial 1	91.3220	
Area 1, Trial 2	91.2124	12.784
Area 1, Trial 3	88.1648	10.868
Area 1, Trial 4	89.6231	18.359
Area 1, Trial 5	79.5753	12.187
Area 2, Trial 1	90.2410	
Area 2, Trial 2	83.7101	
Area 2, Trial 3	70.8966	
Area 2, Trial 4	89.0641	
Area 2, Trial 5	86.9711	7.244
Area 3, Trial 1	91.6099	
Area 3, Trial 2	86.5705	
Area 3, Trial 3	87.4515	
Area 3, Trial 4	87.4842	10.436
Area 3, Trial 5	99.5209	
Area 4, Trial 1	108.0498	
Area 4, Trial 2	86.7694	
Area 4, Trial 3	79.5201	
Area 4, Trial 4	89.7390	12.622
Area 4, Trial 5	90.1836	

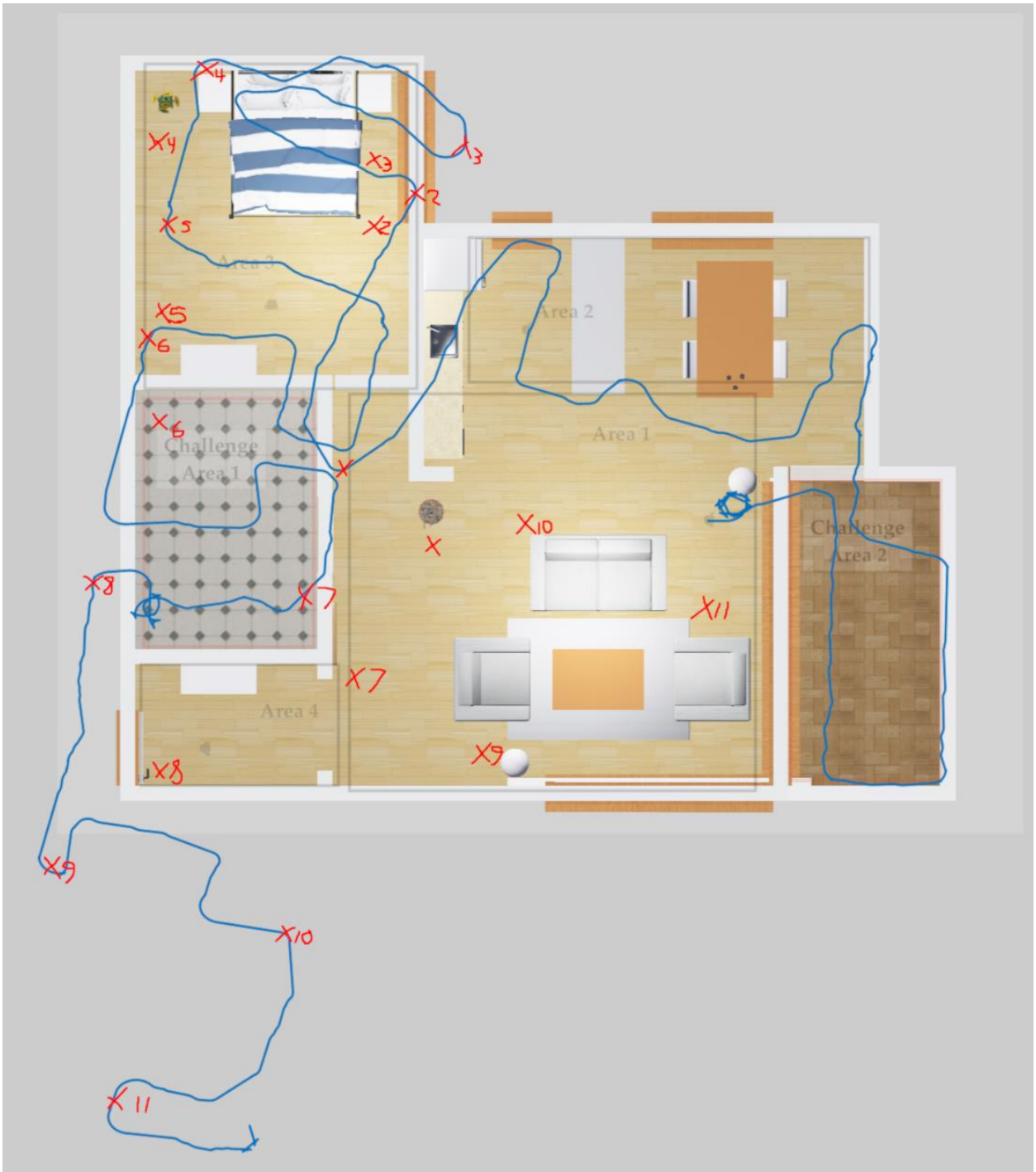
TRIAL PLOTS



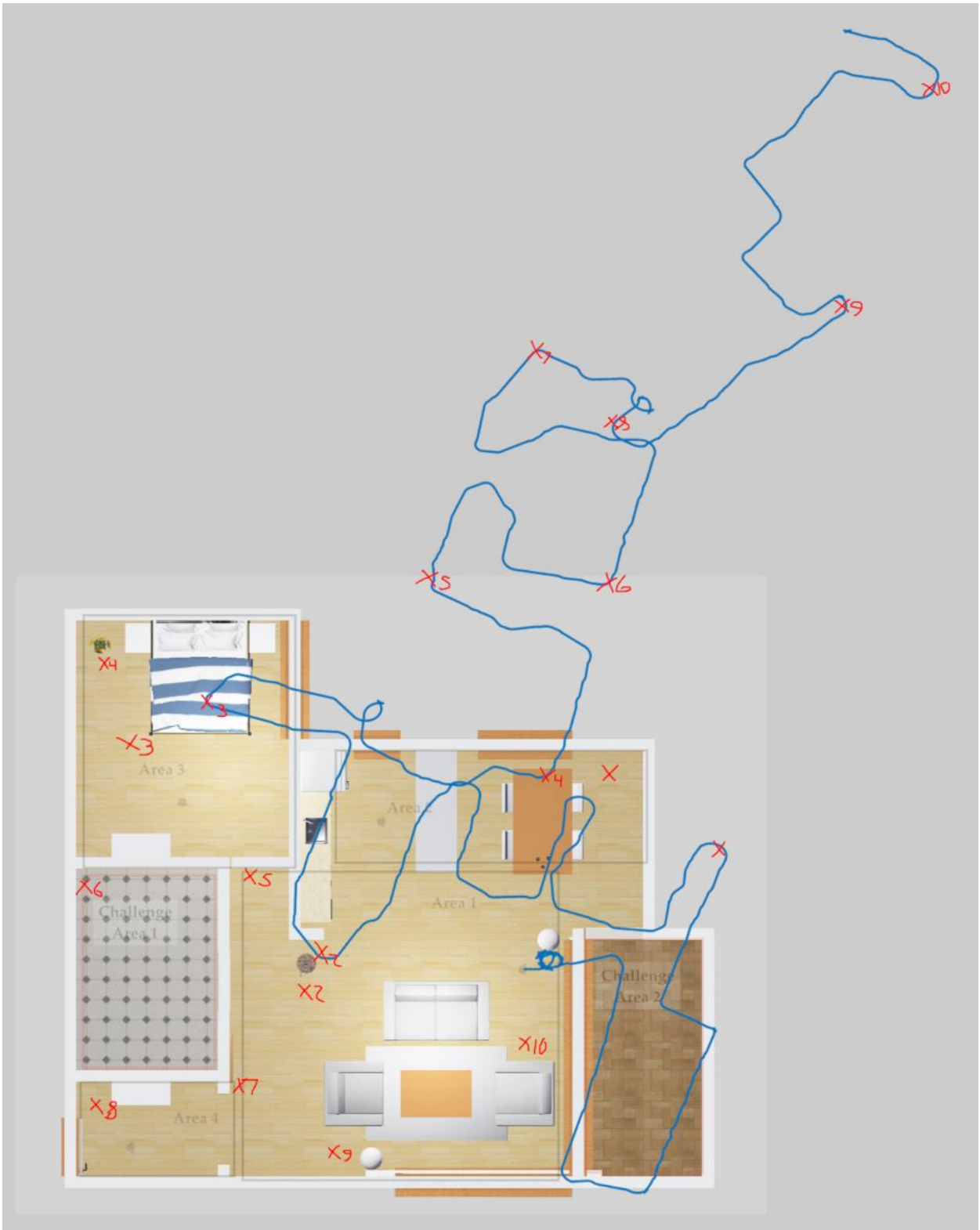
Area 1, Trial 1



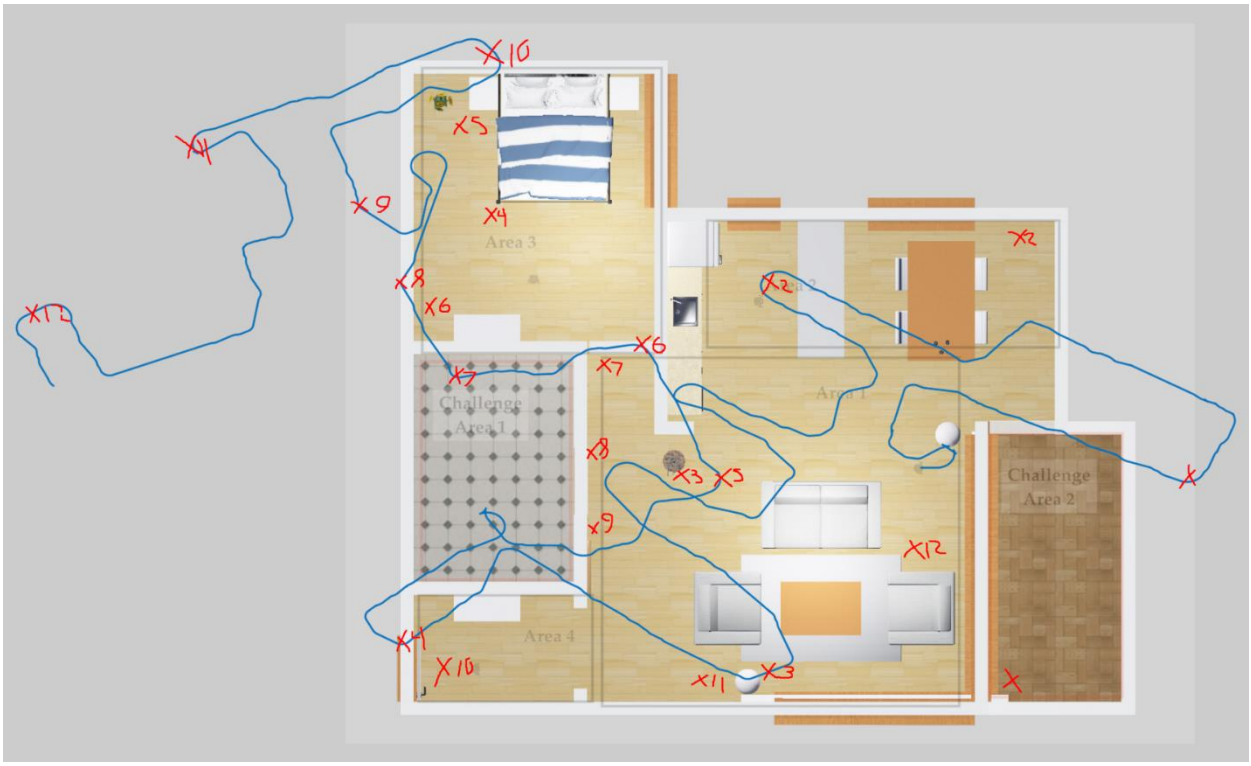
Area 1, Trial 2



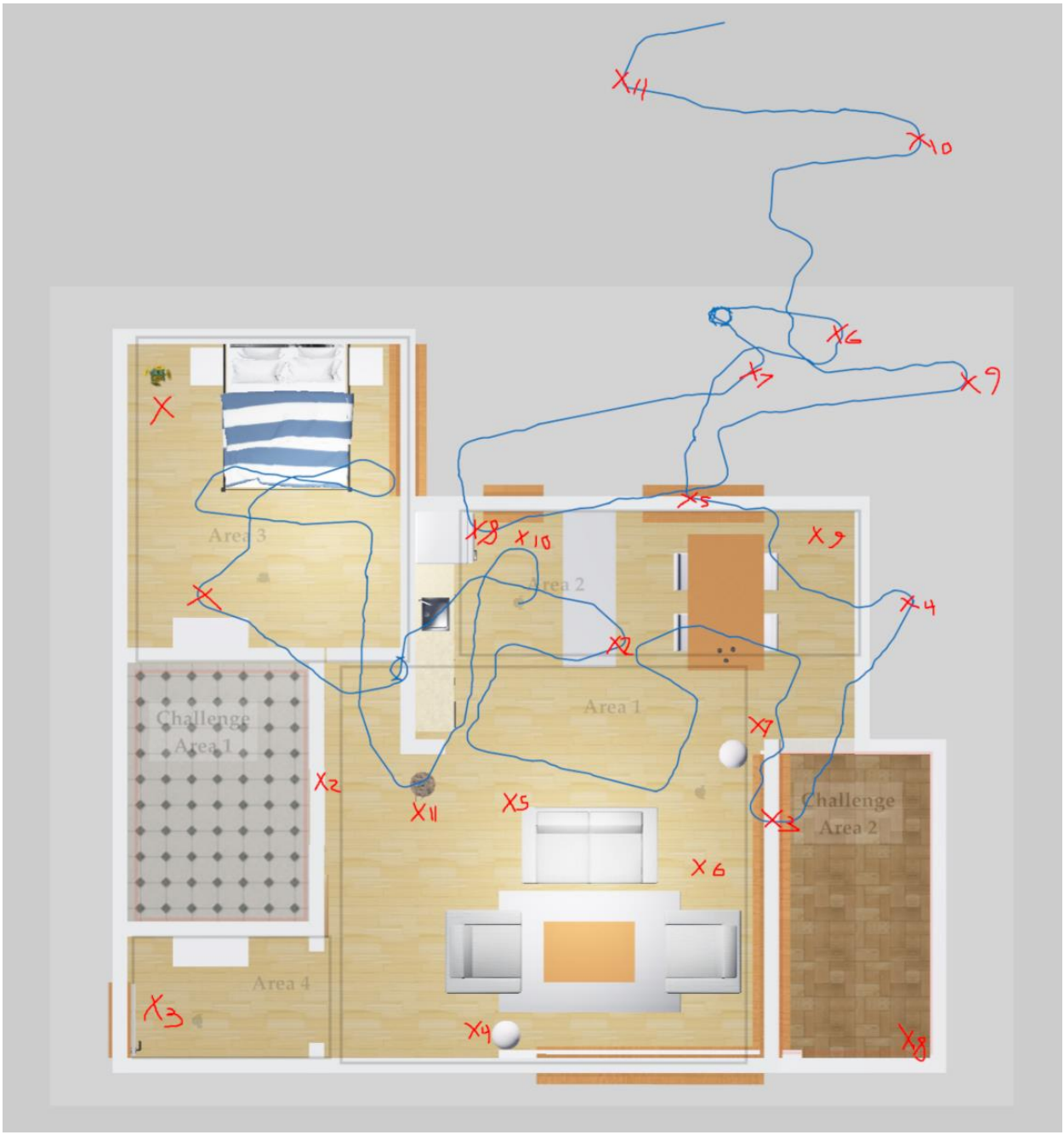
Area 1, Trial 3



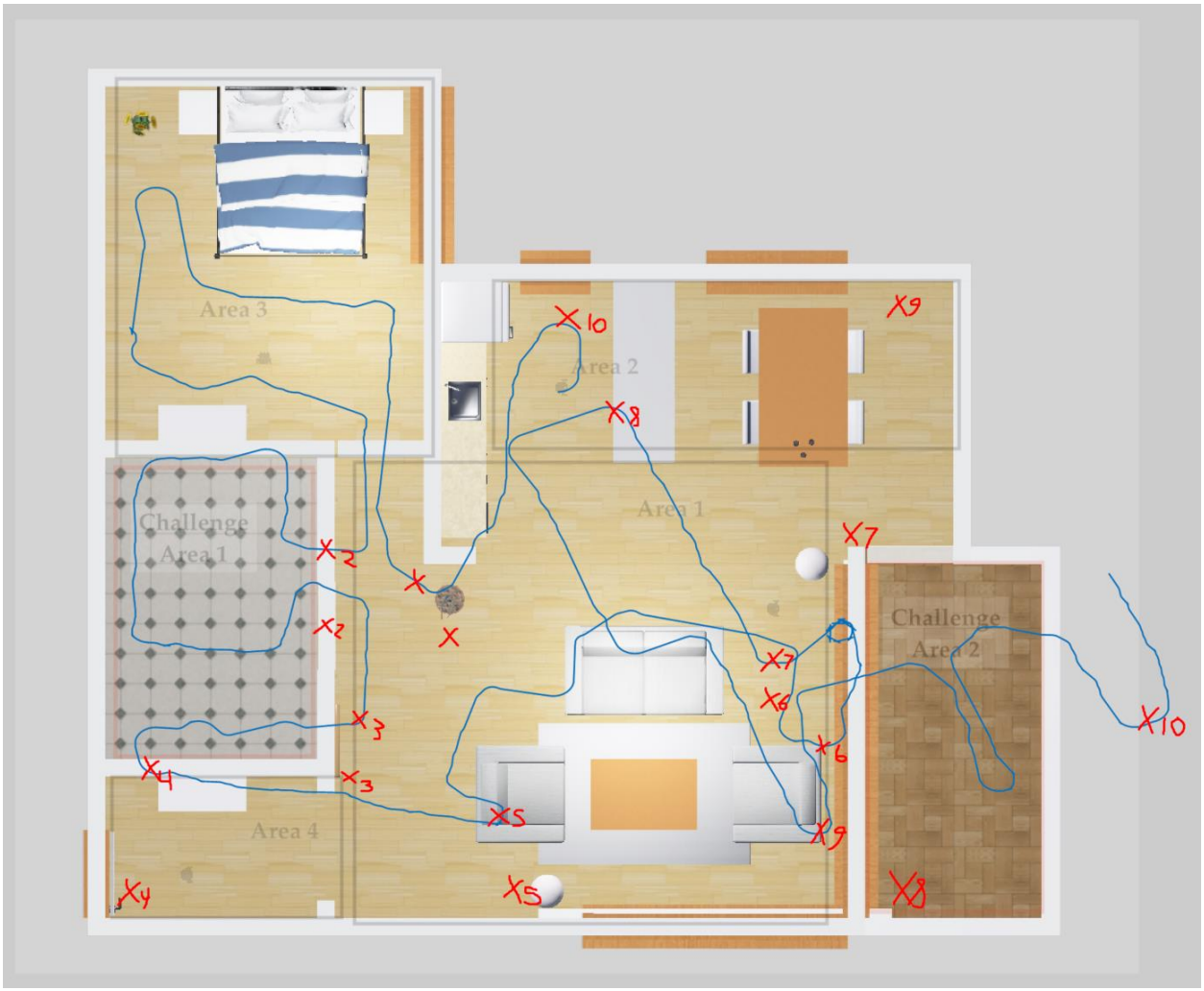
Area 1, Trial 4



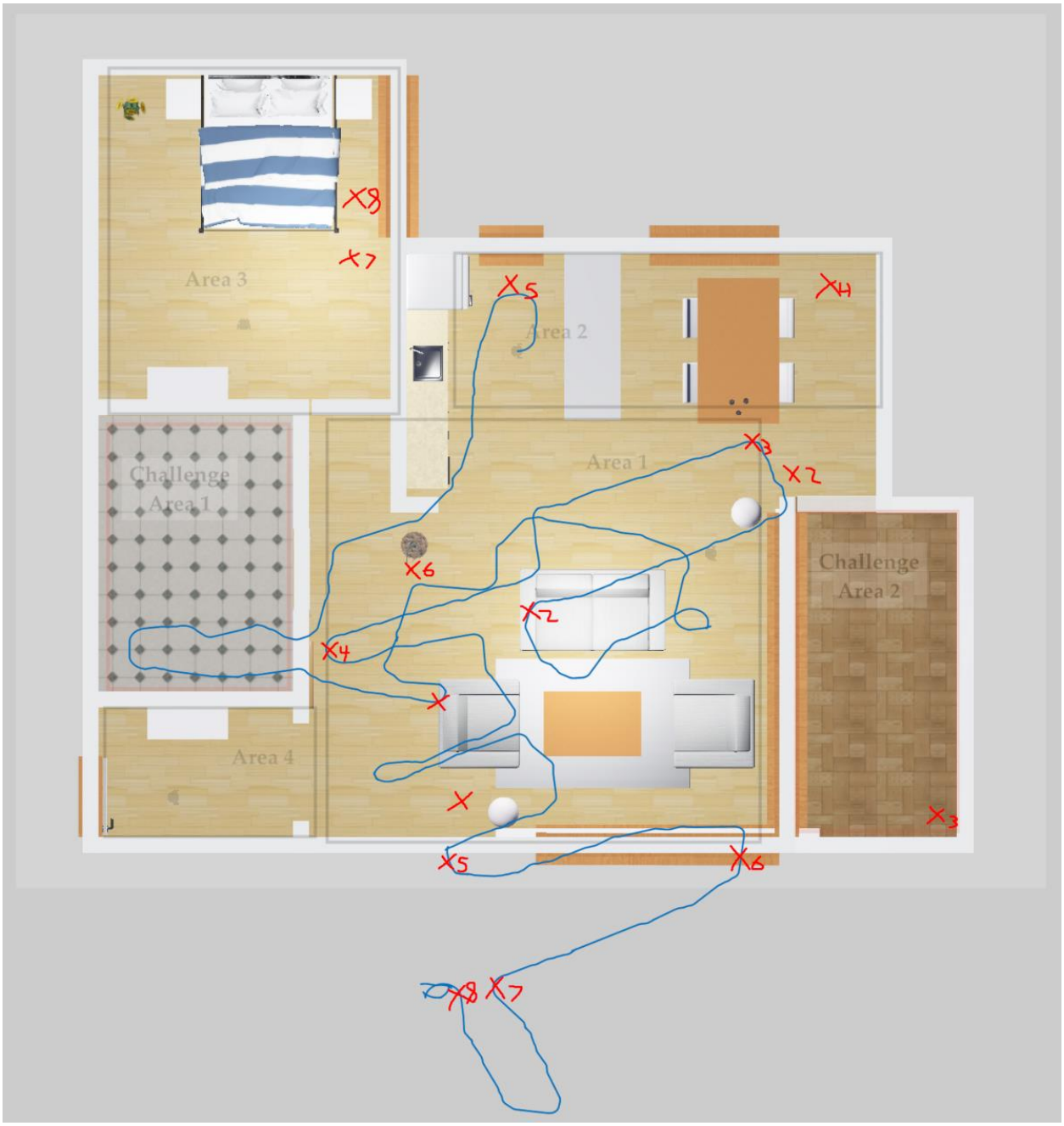
Area 1, Trial 5



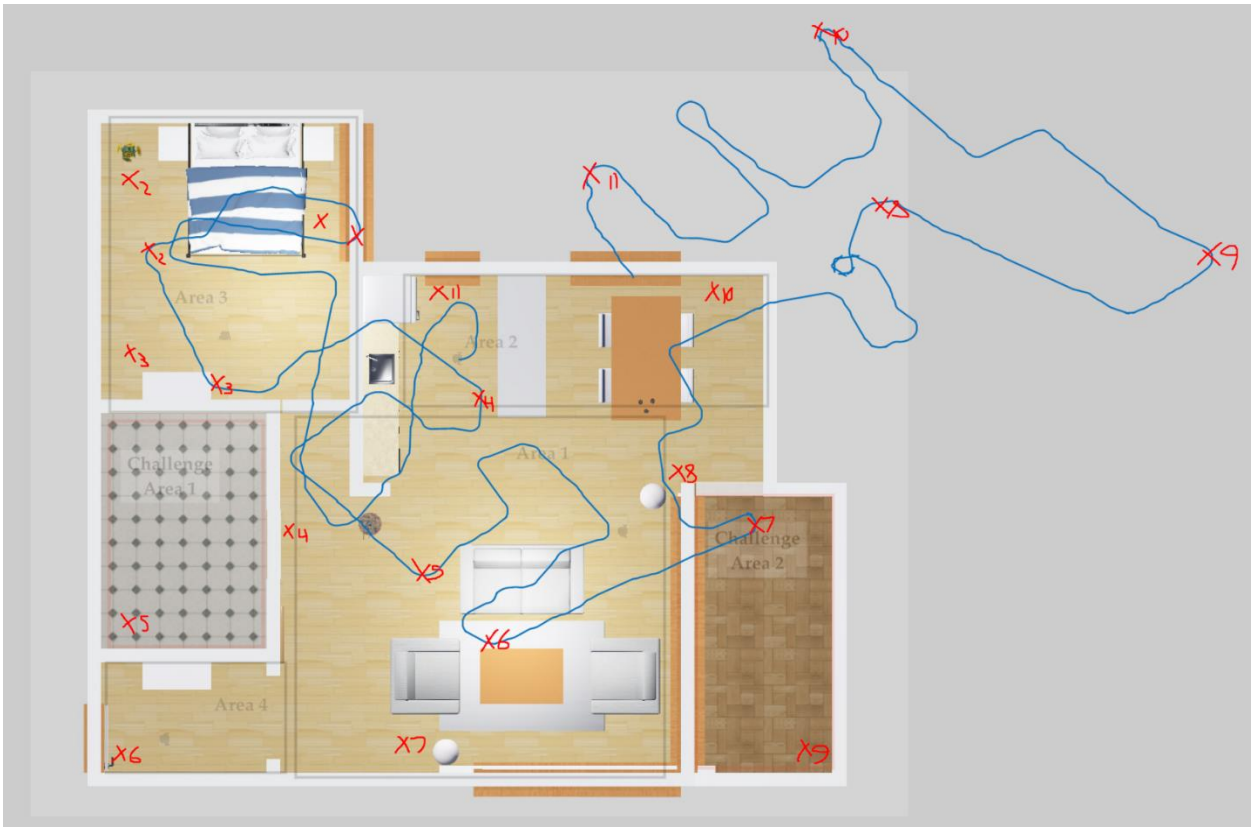
Area 2, Trial 1



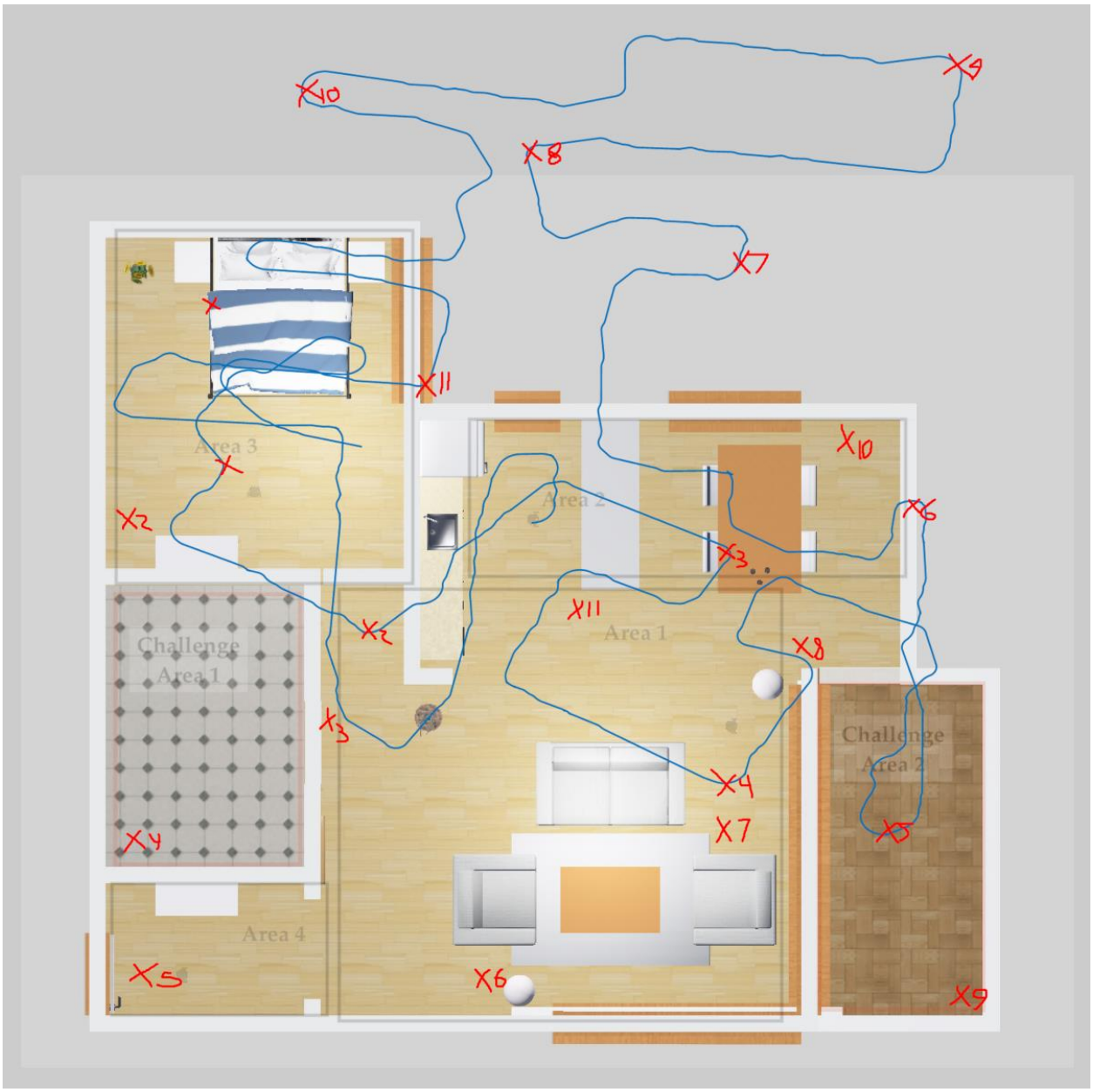
Area 2, Trial 2



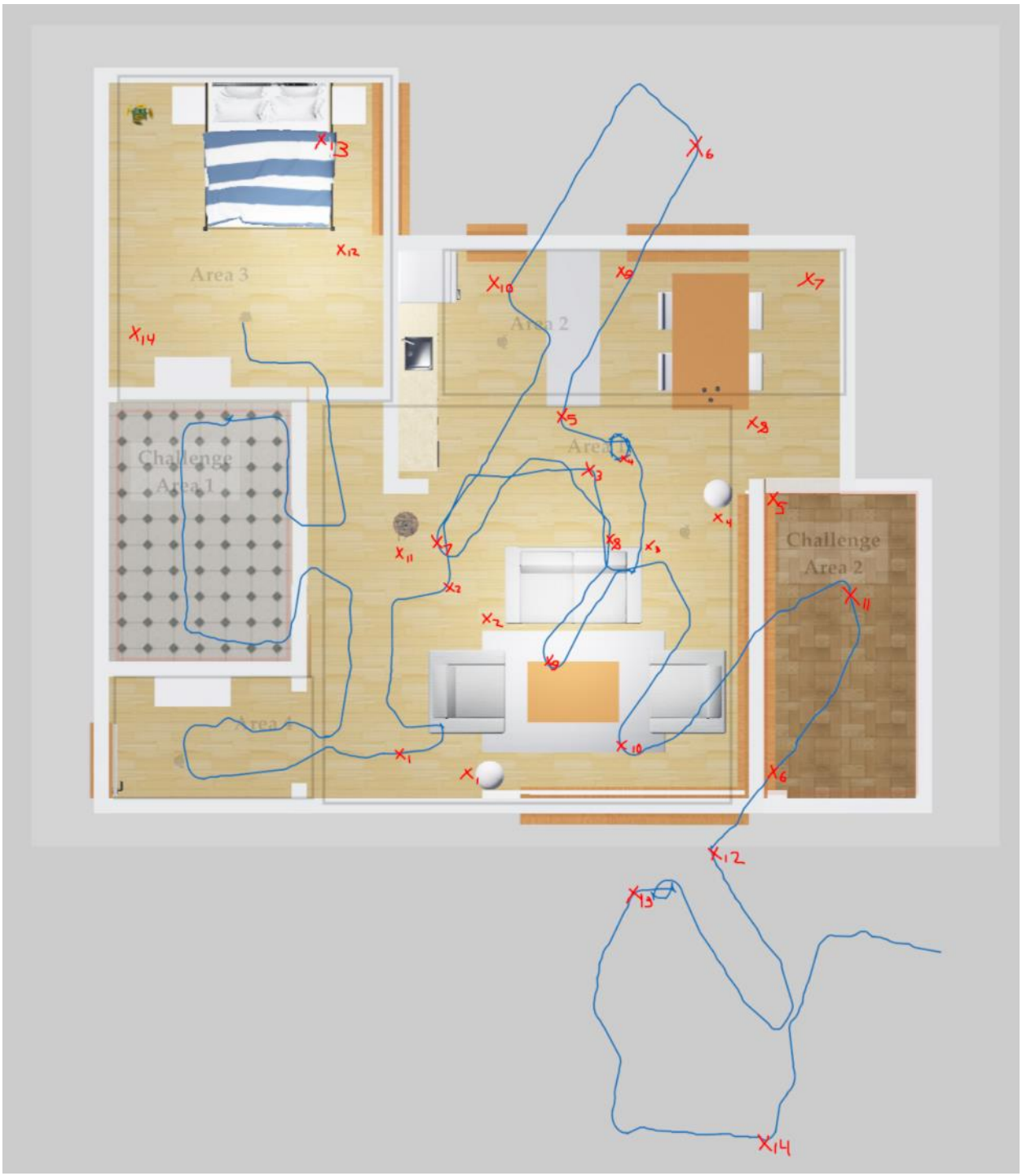
Area 2, Trial 3



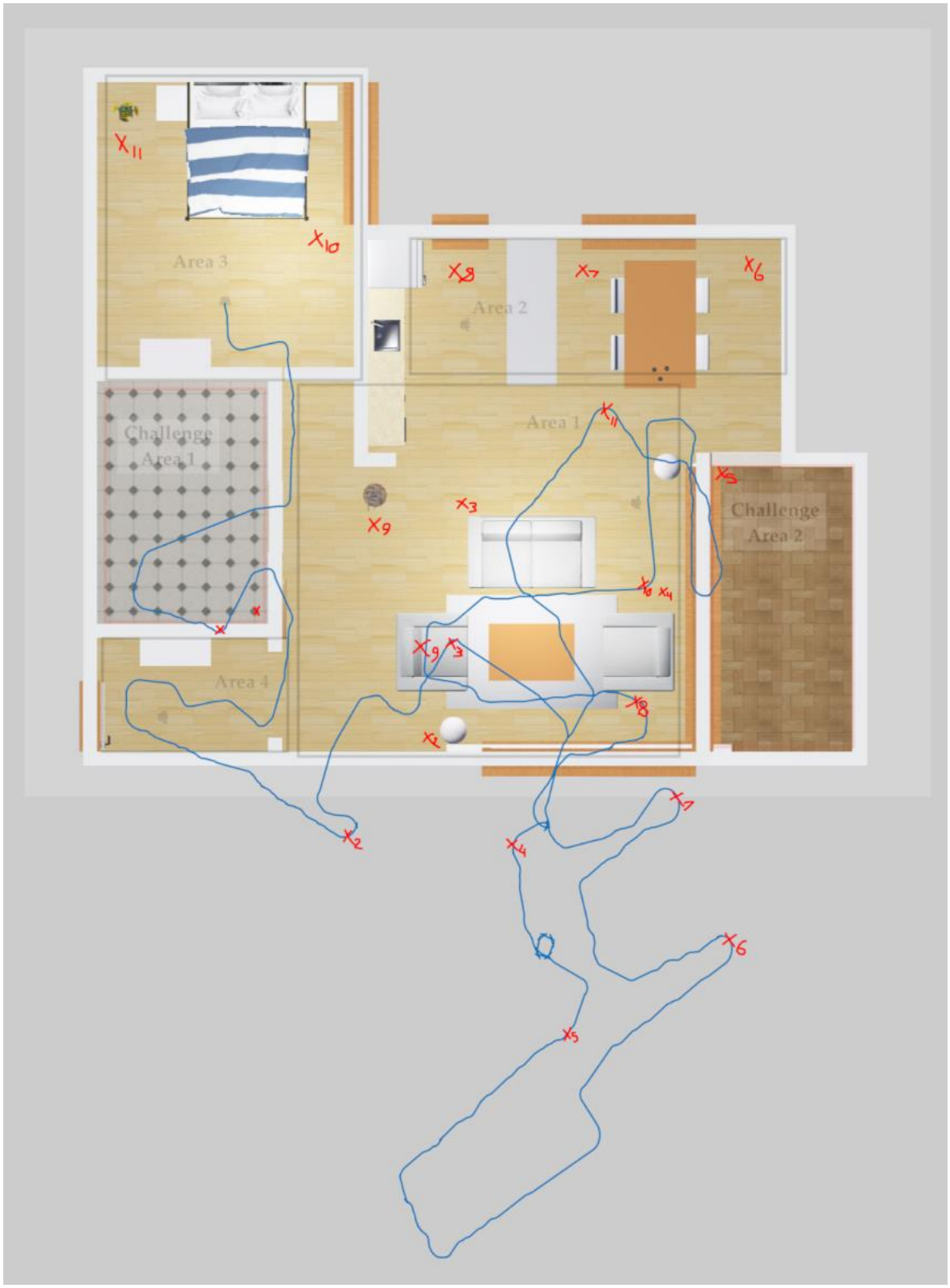
Area 2, Trial 4



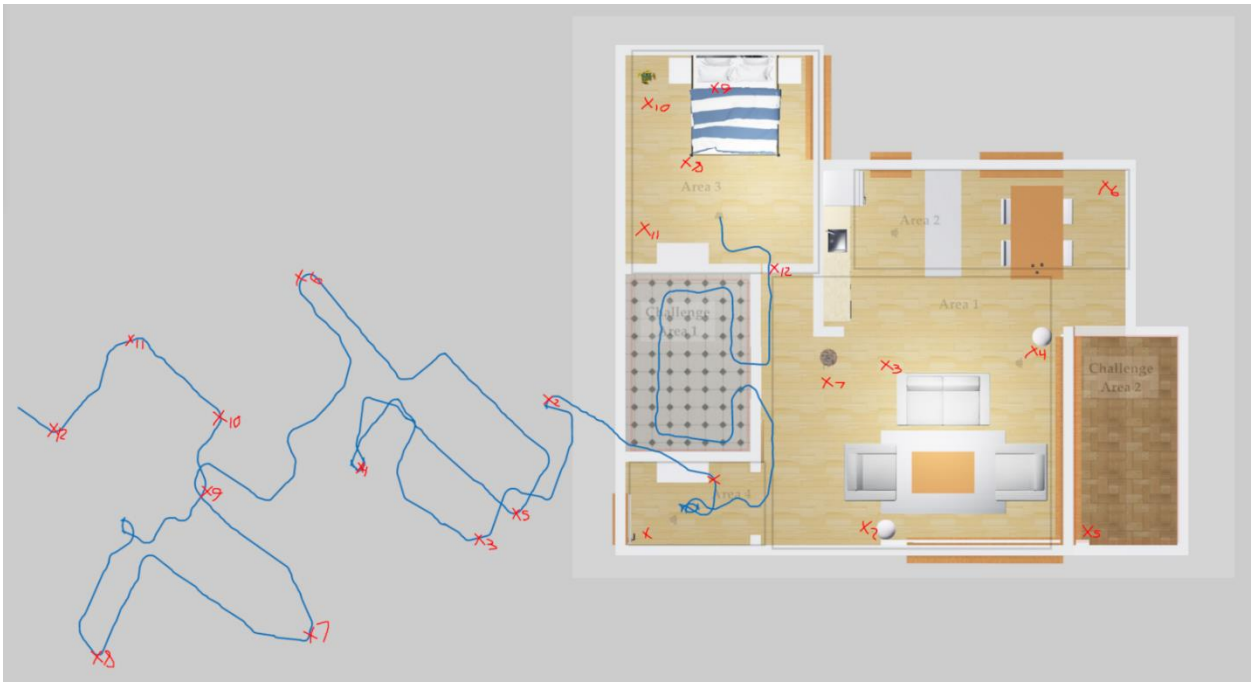
Area 2, Trial 5



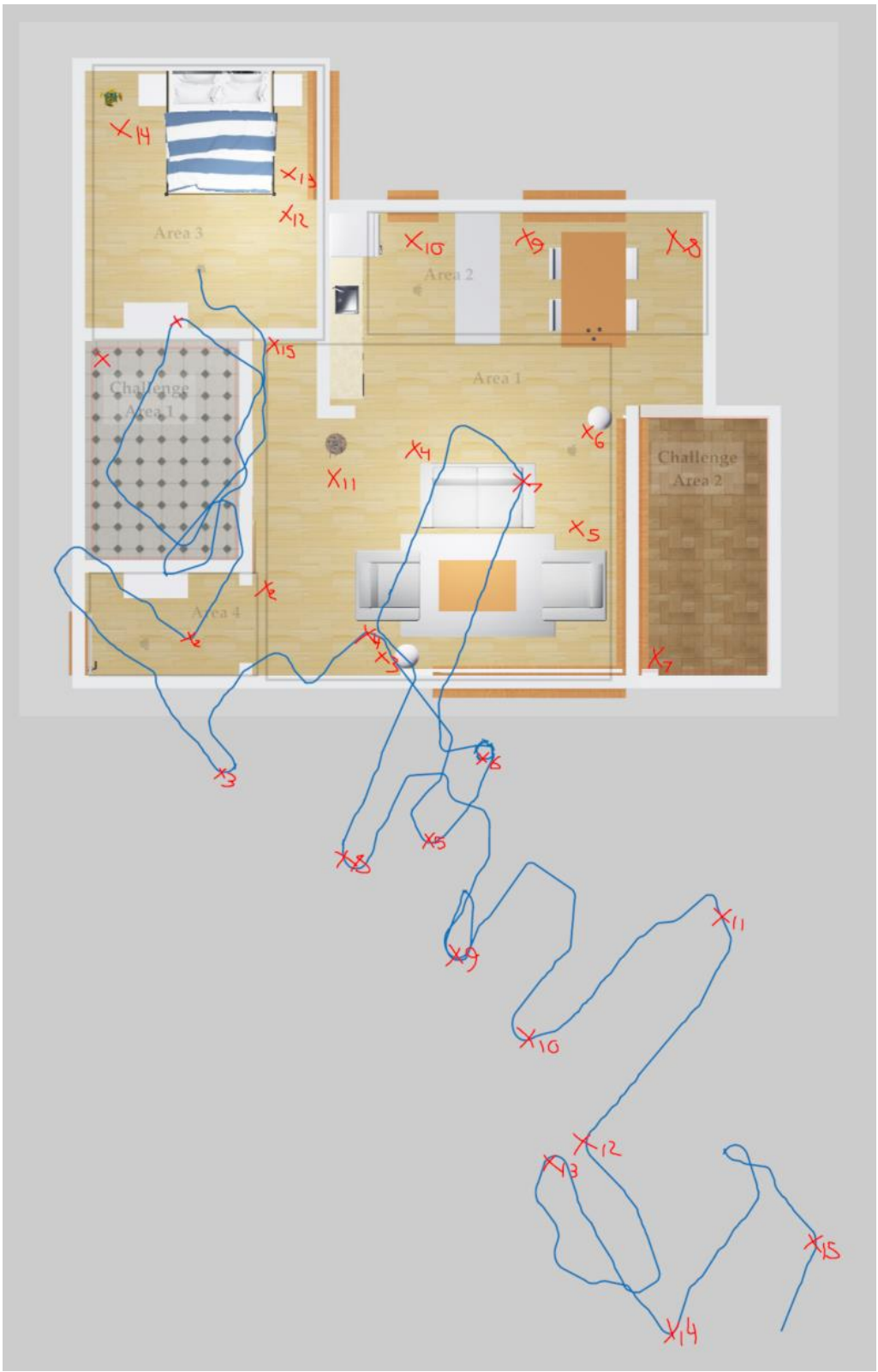
Area 3, Trial 1



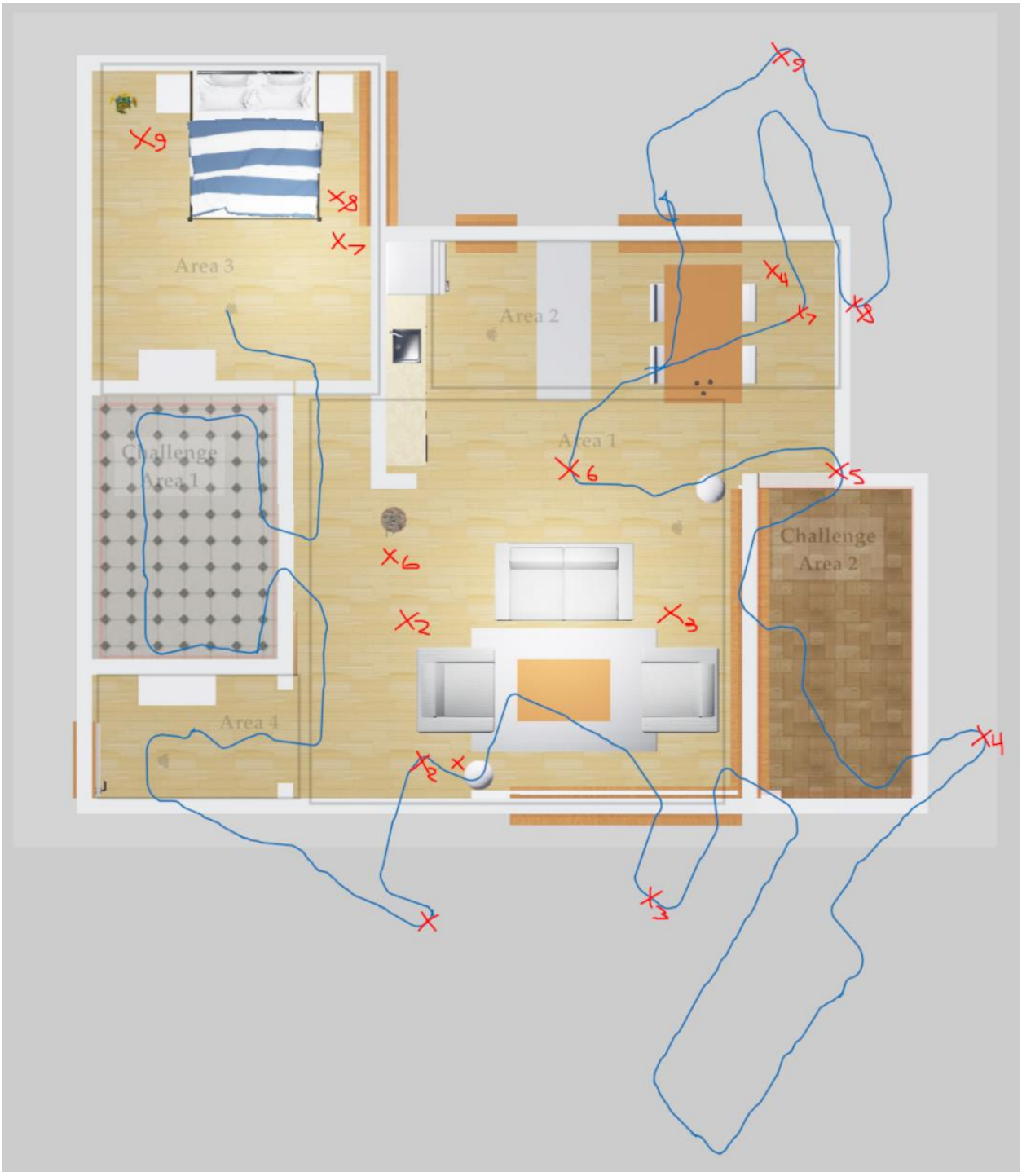
Area 3, Trial 2



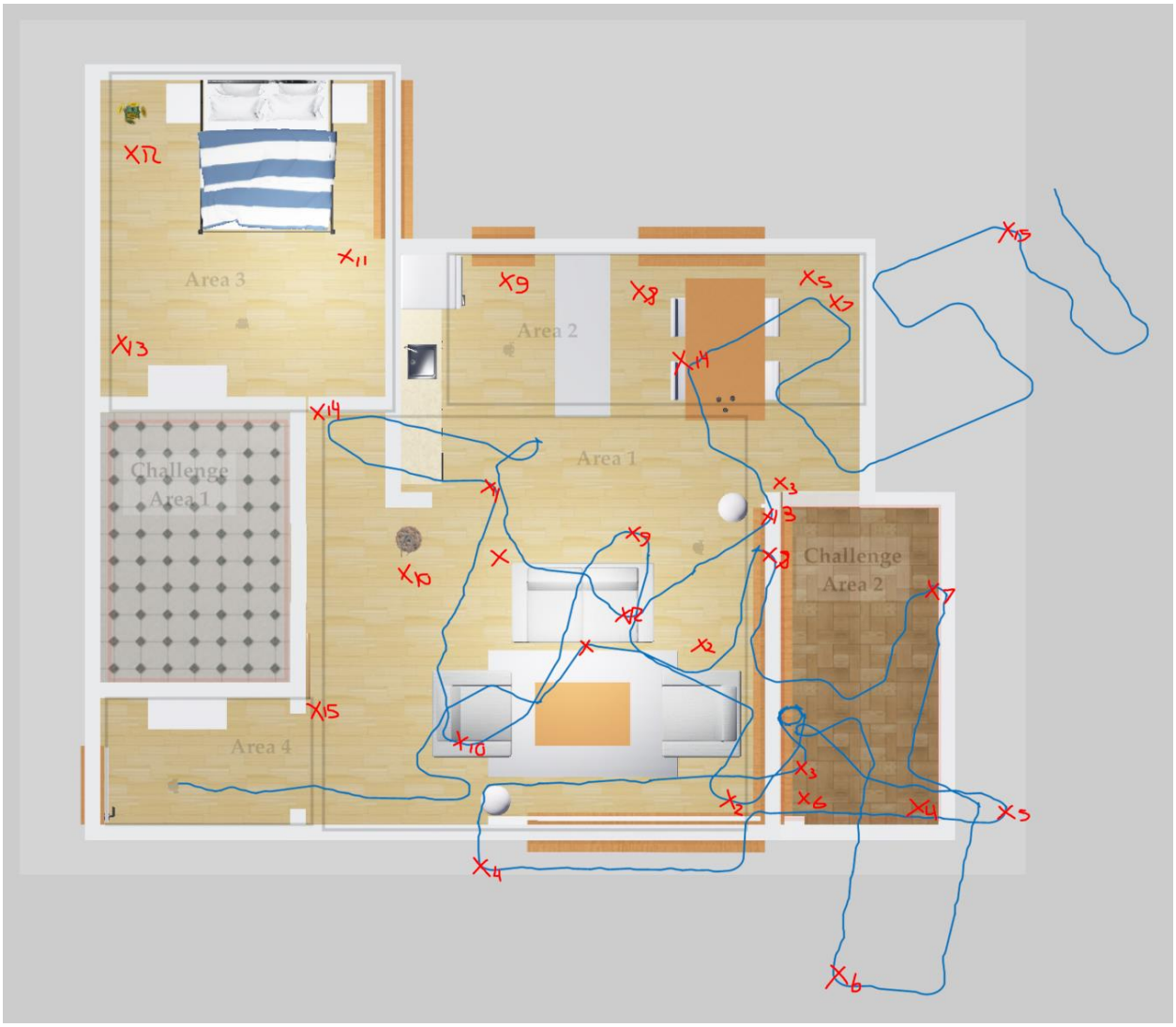
Area 3, Trial 3



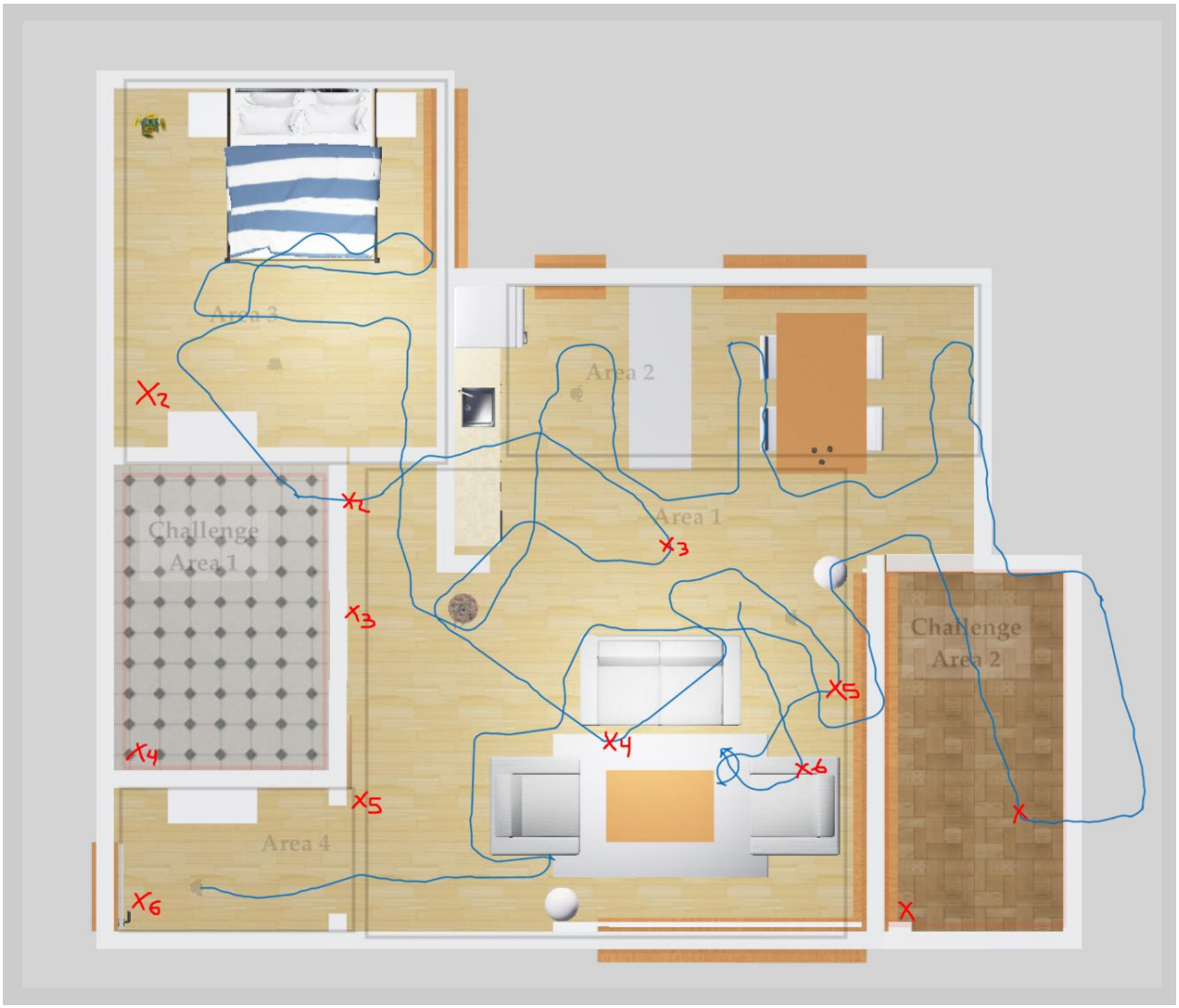
Area 3, Trial 4



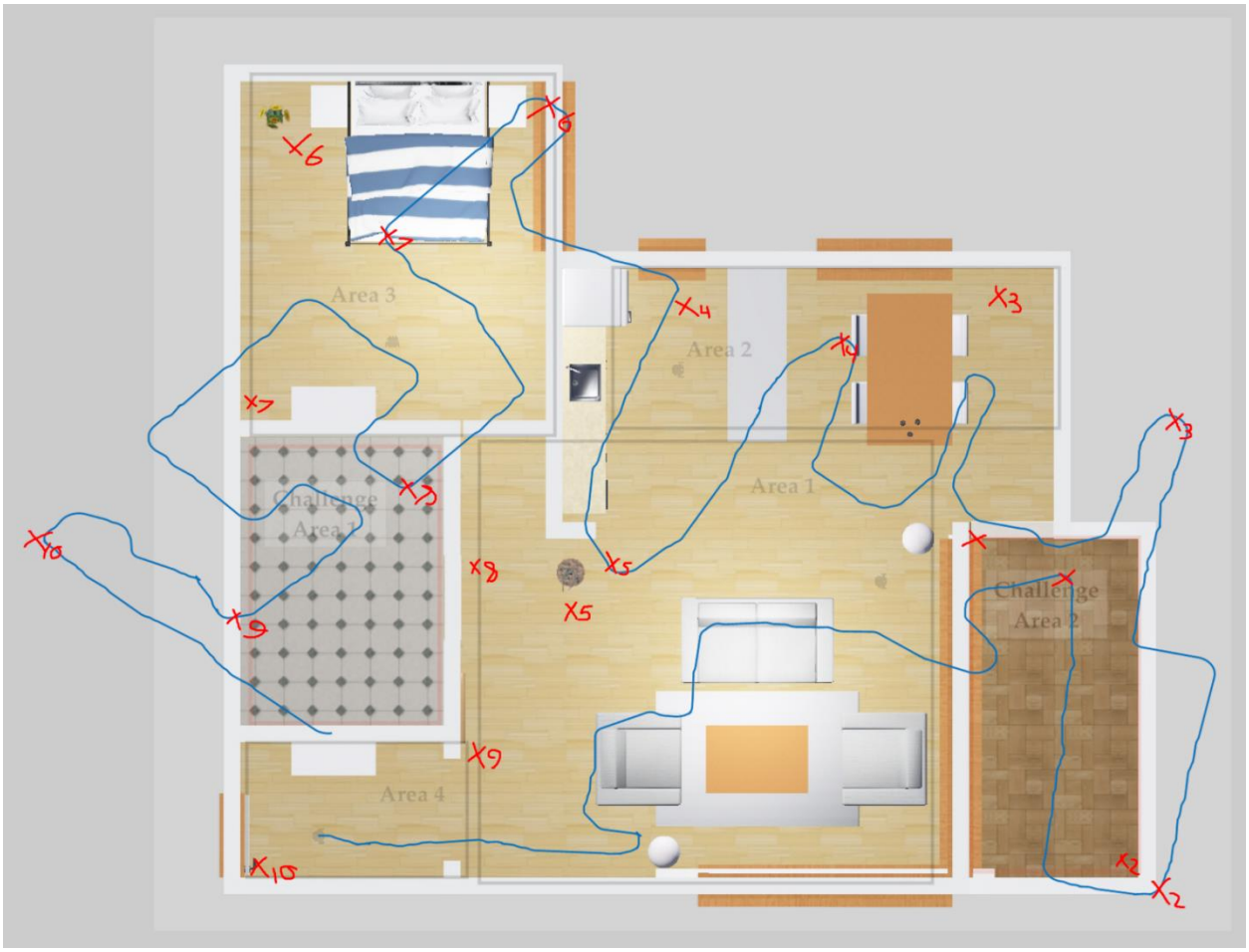
Area 3, Trial 5



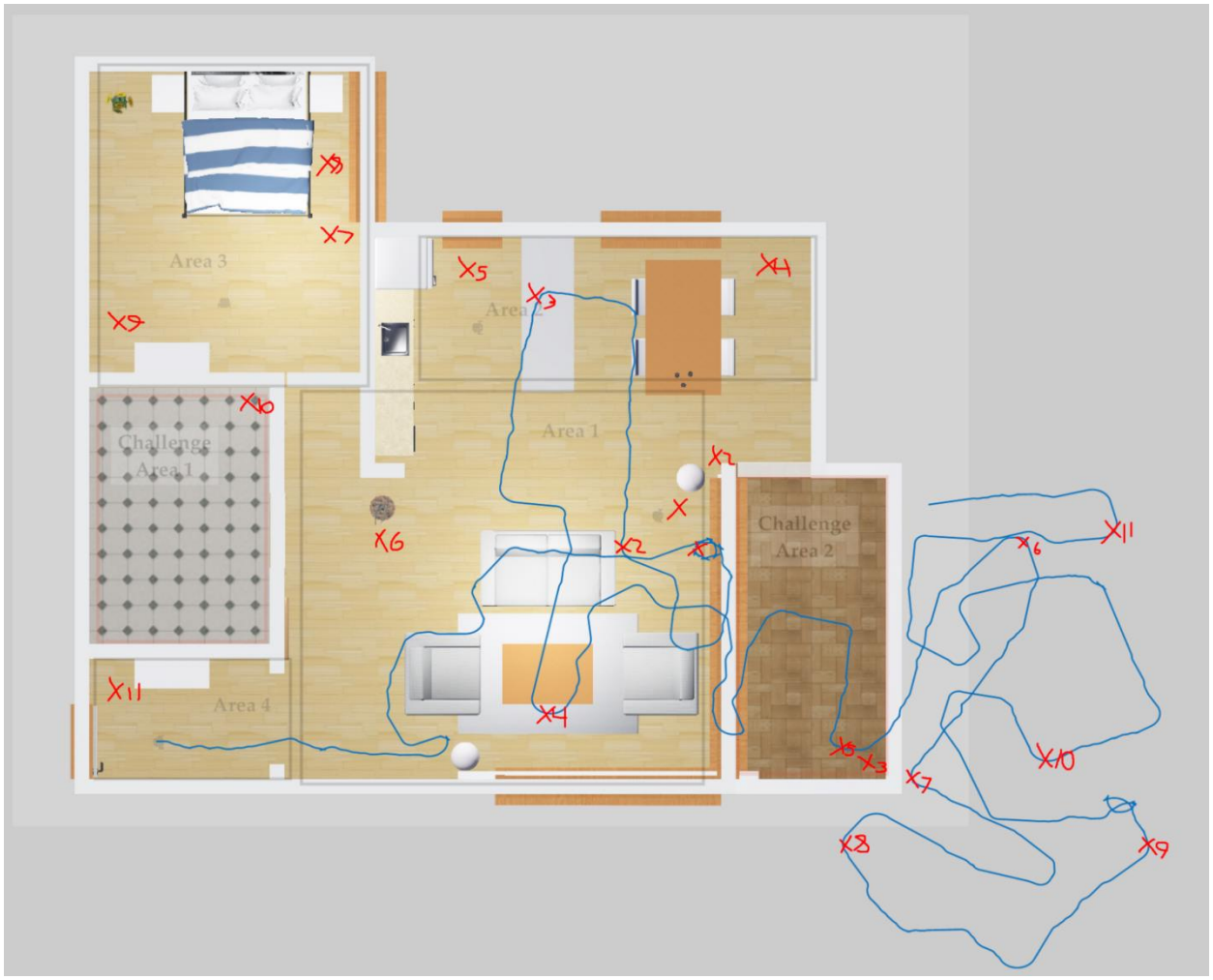
Area 4, Trial 1



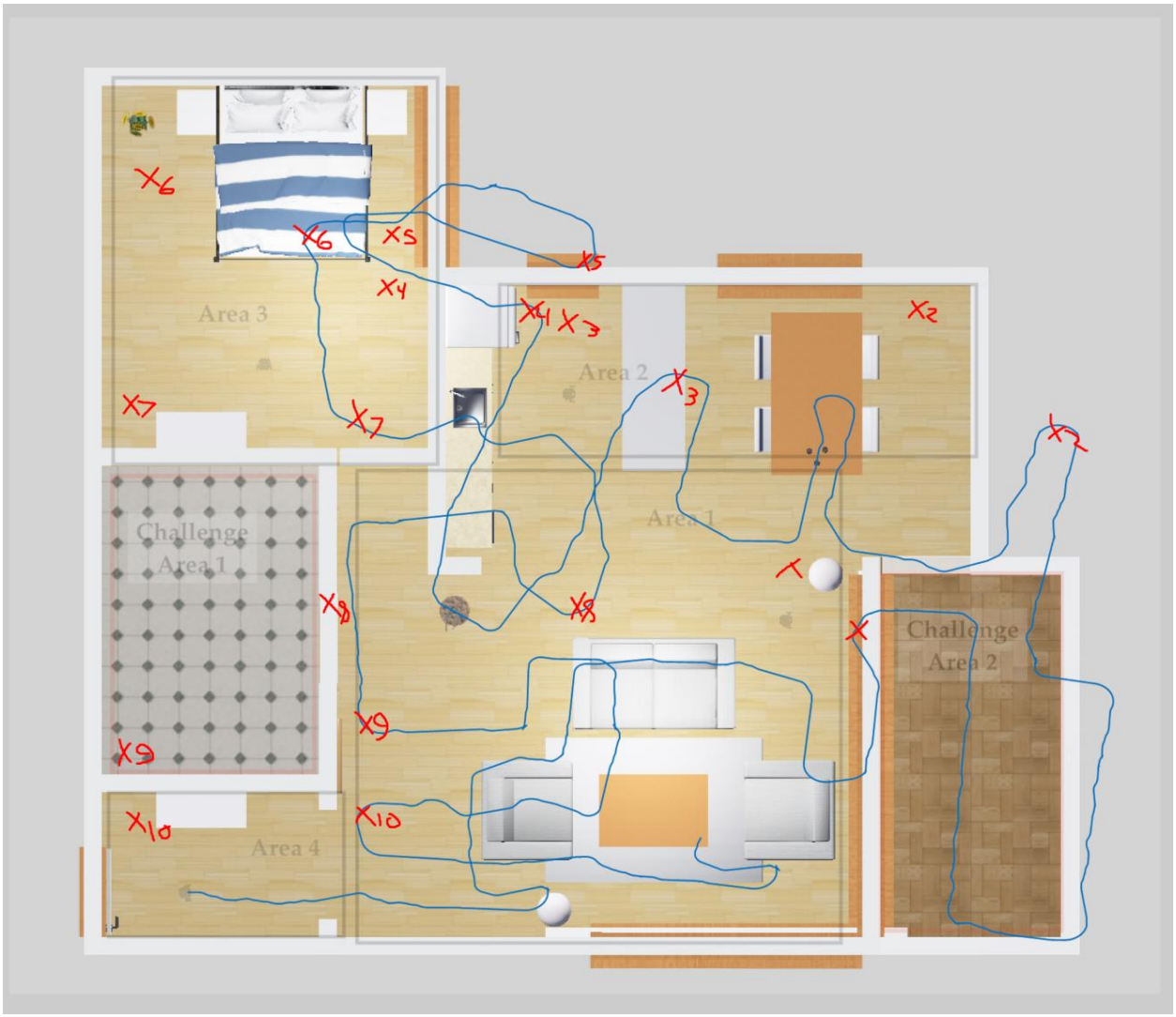
Area 4, Trial 2



Area 4, Trial 3



Area 4, Trial 4



Area 4, Trial 5

For this assignment, we were tasked with writing a controller to maneuver the Turtlebot3 platform around an unknown environment. For my implementation, I took the suggestion from Dr. Monica and implemented a wall-following robot with stall recovery. This was written by hand, using the given RandomWalk code from the assignment as a starting point.

The implementation of the wall-following code uses the initial movement code from the RandomWalk node, adding a check to see if a wall is within sufficient range to the right of the robot. If so, it will turn towards the wall until it is a safe distance away, then it will continue moving forward. If the wall is too close, it will turn away from the wall and continue moving forward. This creates a repeating cycle where, after turning left and right a few times, it will generally move parallel to a given wall. When a wall segment ends, the robot will turn to the right to go around the assumed corner. If a wall is in front of the robot, it will turn left to follow along the given wall. Lastly, if the front sensor value hasn't changed in the past 10 timer runs, the controller will assume that the robot is stalled. When this occurs, it will rotate and slowly back up until the stall is assumed "clear" and attempt to move forward again. Alongside this, if the robot is in a position where a wall is too close for it to move forward, it will also assume it is stalled, and repeat this reverse and rotate maneuver.

In total, this results in a simple controller that can successfully navigate the entire space, tracing an outline around the floor plan of the apartment, even managing to navigate through both Challenge Areas. However, there are a few drawbacks that have been noticed. Firstly, thin objects tend to be troublesome for the robot, such as the lamp along the right edge of Area 1. The robot tends to get stuck on this, but it does manage to detect a stall and recover from this, causing a repeating pattern of getting stuck, noticing a stall, and moving just a bit more until it gets out of this pattern and continues to Challenge Area 2.

Another difficulty with this is that we are not performing any kind of mapping with our LIDAR sensor at the top. As such, the reported position of the robot drifts drastically over time. The overall distance covered is roughly correct, and the starting plot of the robot roughly lines up with where it actually moved, but by the end of the log, the position is vastly different from where the bot actually is after a successful lap, which is near the original starting point of the trial. Adding a rudimentary form of mapping through LIDAR would resolve this issue.