We have been able to write an efficient and compact code which satisfies all requirements from each part of the assignment.

An A* search has been implemented to achieve the task and it keeps track of the oracles that have been already visited to be able to find identity.

In part 4, the challenge of being a dynamic world with constantly moving obstacles, oracles and charging stations has been overcome using the repeat command. This command always returns true and allows you to backtrack and try new solutions indefinitely. Therefore, our solution guarantees robustness as it will try to solve a task up until all the parts of the task (the search and the actual movements can be made. Thus, every time a planned search is not feasible because a movement of the obstacles makes it impossible, the search is rescheduled and reattempted, until the desired task is completed. It may be a bit of a simplistic approach, and it is in terms of code a very tiny modification. However, it does provide consistency. Nonetheless, there might be more dynamic and efficient solutions, at the expense of a longer and trickier code.

To find the identity given to the user, we modified the A* search so that it bans some solutions from being considered. This way, at any point we can find ‘the nearest oracle that has not yet been visited’ (by banning all the visited oracles). This allows our solution to make use of an internal memory of visited oracle, while making only very subtle changes to the code of prior parts.