Technical Round Coding Assignment

1. Task Introduction

Imagine a mobile robot where a user moves the robot manually through an environment and the user has the ability to create waypoints along the way for the robot to visit later when commanded to do so. **The robot also has the ability to create waypoints automatically after traveling a certain distance or if a turn is detected as the robot is manually moved. If the robot detects that it is nearby to a waypoint that was previously created within a certain position threshold, no new waypoint should be created. The existing waypoint itself should be used.**

Once the user is done with moving the robot, the set of waypoints that was manually created by the user or automatically by the robot should be persisted to disk and should be able to be read back.

Now, if the user sends a command to visit a particular waypoint from the set waypoints, **the robot should only move through the set of known waypoints to reach the destination waypoint while keeping the number of waypoints visited or distance traveled to a minimum**. The intermediate plan to reach to successive waypoints can deviate from how it was originally driven during the waypoint creation. This ensures that any obstacle introduced between two waypoints can be successfully navigated around if required.

2. Deliverables

The following points are expected to be covered in order for the task to be considered as completed.

- A package written in **ROS 1** that is able to achieve the task described in section 1 with nodes and lunch files.
- The package should be written in **C++** with the help of classes and any other concepts that are deemed necessary. The code should be readable and easily maintainable.
- **The performance and scalability of the implemented package will be considered.** It should have the capability to handle a large set of waypoints.
- All the waypoints that are manually or automatically created should have a unique ID associated with its pose (x,y, theta) on the map.
- **Services to create the waypoint manually by giving it a unique ID.** The waypoint should be created where the robot is currently on the map.
- **The waypoints should be visualized using normal markers or interactive markers in ROS 1 in RViz.** Make sure the set of waypoints that was manually created by the user is visually distinguishable from the set of waypoints that was automatically created by the robot. If interactive markers are used, it would be easier to give goals to reach a

destination. For example, using button or mouse clicks on the interactive marker can be used to set the goal or destination waypoint the robot should reach. (This is optional)

- **A video demonstrating the implemented package**, where a goal is given and the robot traverses through the set of known waypoints to reach the final destination or goal waypoint as described in section 1.
- **The implemented package along with a README file should be included which describes how the package can be used and how it works.** The package with the README file can be compressed and shared via email to anscersoftwarerecruitment@gmail.com.

3. Support and Hints

For the simulation of the robot either ANSCER's simulation of AR100 robot can be used or Turtlebot simulation can be used as well. The links for the simulations are provided below.

- For ANSCER AR100: https://wiki.ros.org/AnscerRobotics/AR100
- For Turtlebot3 Burger: https://wiki.ros.org/Robots/TurtleBot

The following hints or assumptions can be used in the implementation of the required package.

- The robot can be assumed to always be on top of a known waypoint.
- Using the pose of the robot in the map, the current waypoint the robot is on can be determined. For example, if the robot is within 0.5 meters of a known waypoint. The robot can be assumed to be on top of that particular waypoint.
- For persisting the waypoints and the relationship between them to disk either JSON or YAML format can be used.
- For the path planning between waypoints, the default global planner that is used in the `move_base` node is enough.
- Once the order in which the nodes should be traversed is determined, the goals can be successively given to the `move_base` node using the default action server implemented with it. The robot might stop when a new goal is given. This issue can be ignored as this is caused by the `move_base` node implementation.
- In section 1, the robot is said to have the capability to automatically create waypoints after it has moved a certain distance or turn angle threshold. The distance threshold can be 2 to 3 meters and the turn threshold can be between 45 to 90 degrees if it is decided to be used.
- Custom ROS 1 messages and service can be created to be used in the package nodes.