# IVDC - Automathon’24

# Autonomy Team Challenge

PATH PLANNING ALGORITHMS:

Task 1:

1. Dijkstra's Algorithm: A classic algorithm for finding the shortest path between nodes/vertices. It's a single-source shortest path algorithm based on breadth-first search.

2. A\* Algorithm: An extension of Dijkstra's algorithm that introduces heuristics to guide the search, making it more efficient by focusing on the most promising paths.

3. D\* Algorithm: An incremental search algorithm that dynamically updates the path based on changes in the environment, making it suitable for scenarios where the map is not fully known or may change over time.

4. Rapidly-exploring Random Trees (RRT): A probabilistically complete algorithm used in robotics for finding a feasible path in high-dimensional spaces. It grows a tree in the configuration space of the robot.

5. RRT\* (Rapidly-exploring Random Trees Star): RRT\* is an extension of the RRT algorithm that optimizes path cost by periodically rewiring the tree, resulting in more efficient and often more optimal paths for robotic motion planning.

These algorithms can be classified based on their approach as deterministic (Dijkstra's, A\*, D\*), probabilistic (RRT, RRT\*), and heuristic-based (A\*). Each has its strengths and weaknesses in terms of optimality, efficiency, and applicability to different scenarios.

Task 2:

For an Unmanned Ground Vehicle (UGV) navigating a track with various obstacles while requiring speed and computational efficiency, the RRT\* algorithm would be a suitable choice because:

* Optimal Paths: RRT\* algorithm improves path optimality compared to RRT by periodically rewiring the tree to optimize path costs.
* Efficient Exploration: It efficiently explores complex spaces, quickly generating feasible paths through obstacles like trees, shrubs, and street furniture.
* Adaptability: RRT\* adapts to dynamic environments, continuously refining paths as new information becomes available, ensuring effective navigation even in changing conditions.
* Computational Efficiency: Despite additional computations for rewiring, RRT\* remains computationally efficient, suitable for real-time UGV navigation.
* Balance of Speed and Quality: It strikes a balance between finding near-optimal paths and being fast, making it ideal for navigating tracks with diverse obstacles while meeting speed requirements.

Task 3: On Github: <https://github.com/MitanshuKumawat/IVDC_Induction.git>

ROS BASED PS:

ROS (Robot Operating System) is an open-source framework for developing robotics software. It provides libraries and tools to help developers create complex robot applications more efficiently. ROS facilitates interoperability, modularity, and reusability of code, enabling collaboration among researchers and industry professionals. Major ROS distributions include Melodic Morenia, Noetic Ninjemys, and Foxy Fitzroy, with each offering improvements and new features to support evolving robotic applications.

Task 1:

The sensors and other electronic components used in the 'smb' robot are as follows:

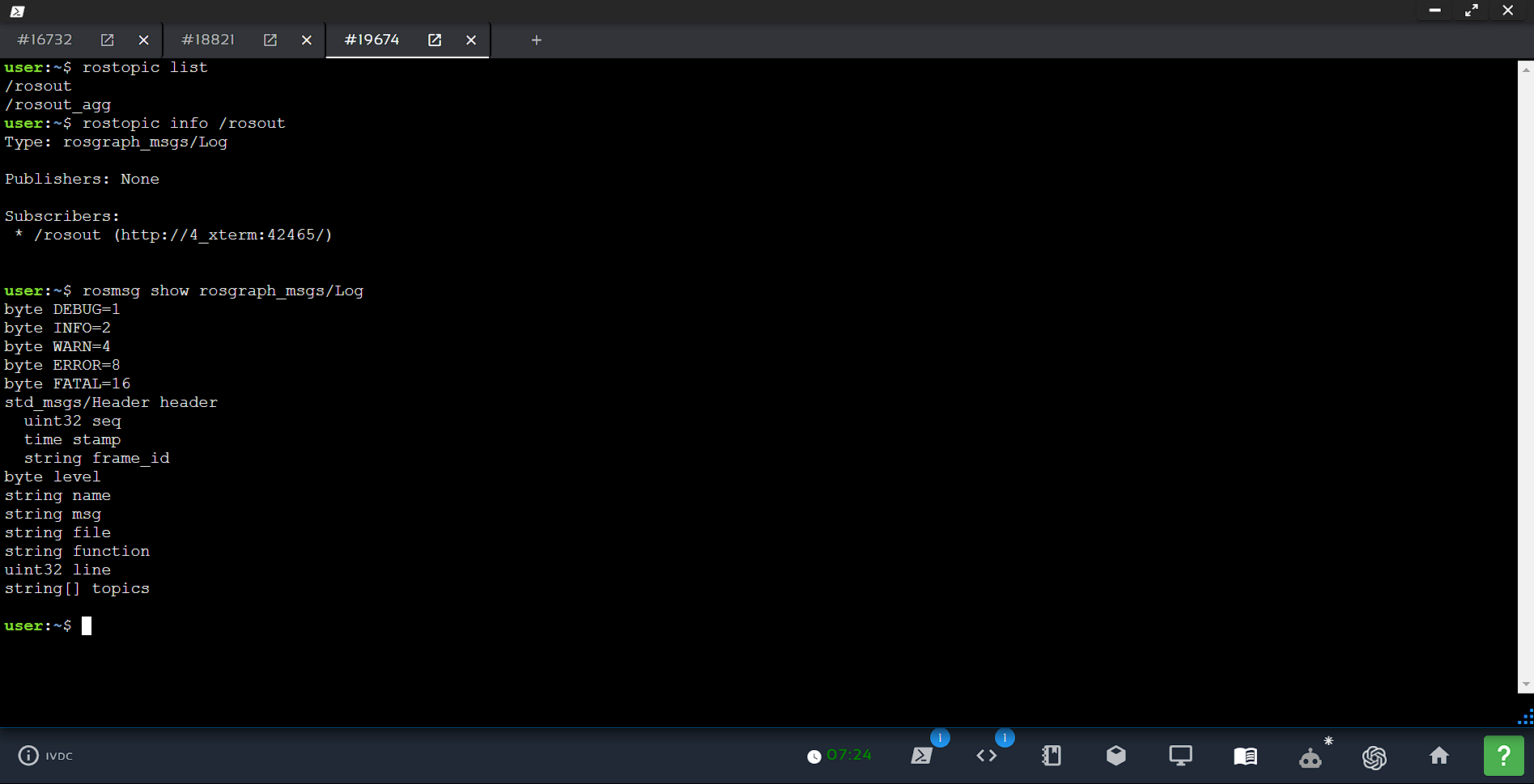
1. Blackfly camera
2. Depth sensor
3. GPS module
4. Inertial Measurement Unit (IMU) sensor
5. Monocular camera sensor
6. LIDAR sensor

The file in the workspace that provides information about these sensors and the robot type is likely named 'smb\_description.rviz'. Additionally, information about sensor plugins can also be found in the folder named 'sensor\_plugins'. This file and folder likely contain configuration and implementation details related to the sensors used in the 'smb' robot.

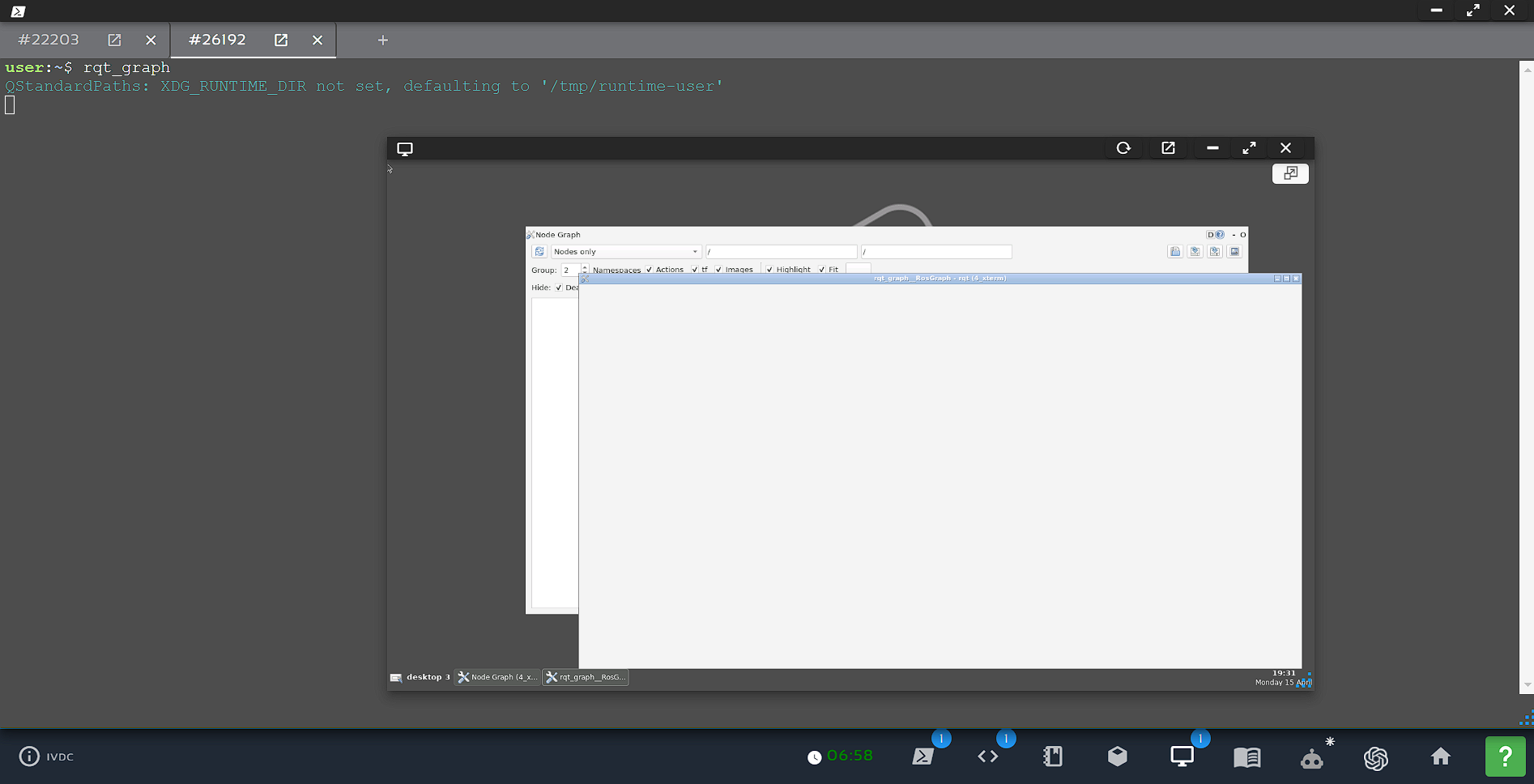
Task 2:

1. Service: It's like making a request and getting a response, where one node asks for something and another node provides it.
2. Action: Similar to a service, but it's used for longer tasks or tasks that need continuous updates.
3. Messages: These are the bits of information that nodes send to each other. They're like the language nodes use to talk to each other.
4. Nodes: Think of them as individual workers. Each one does a specific job, like reading sensors or moving motors.
5. Topics: These are like channels where nodes send and receive messages. They're used to share data between different parts of the system.
6. rqt\_graph: It's a tool that shows a picture of how all the nodes and topics are connected in the system. It helps you see the big picture of how everything works together.

Task 3:



Task 4:



ESTIMATION AND LOCALISATION:

Task 1:

A point cloud data structure is a collection of data points in space, most commonly in a three-dimensional coordinate system. Each point in a point cloud represents a single sample on the surface of an object or space.

**Coordinate Transformation**: The raw LIDAR data is typically recorded in the sensor's own coordinate system, which may not be directly usable for further processing. The first step is to transform the coordinates of the measured points into a common, global coordinate system, such as the world coordinate system or a local coordinate system defined by the user.

**Point Cloud Creation**: Each LIDAR measurement can be represented as a single point in a three-dimensional (3D) space, where the x, y, and z coordinates of the point correspond to the measured distance in each dimension. These individual points are then grouped together to form a point cloud, which is a collection of 3D points representing the spatial distribution of the measured objects.

**Data Structures**: The point cloud data can be stored in various data structures, depending on the specific requirements of the application. One common representation is a simple list of 3D points, where each point is represented as a tuple or a list containing the x, y, and z coordinates. Alternatively, point clouds can be stored in more specialized data structures, such as k-d trees or octrees, which can improve the efficiency of spatial queries and operations.

**Additional Information**: In addition to the 3D coordinates, the point cloud data may also include other information associated with each measurement, such as the intensity of the reflected laser pulse, the timestamp of the measurement, or the color or reflectance properties of the object being measured.

**Code available on github**: <https://github.com/MitanshuKumawat/IVDC_Induction.git>

Task 2:

The precautionary measures to take when testing a LIDAR-based algorithm in a controlled environment:

****Controlled Lighting Conditions****: Ensure consistent and stable lighting throughout the testing process by using dedicated lighting fixtures and shielding the test area from external light sources. Variations in lighting can lead to inconsistent reflections and errors in LIDAR measurements.

****Surface Reflectivity Considerations****: Carefully select and arrange surfaces with known, uniform reflectance properties to minimize the impact of specular reflections and absorption on LIDAR measurements.

****Obstacle Interference****: Eliminate any movable objects or obstructions that could interfere with the LIDAR's line of sight during the testing process. Carefully position the LIDAR sensor and target objects to avoid unexpected obstructions.

**NAME: Mitanshu Kumawat**

**ROLL NO.: 230008022**