University of Technology Sydney

Robotics Studio 2 - Sprint 3

Technical Documentation

Authors:
Hallie Robins - 14253583
Issy Pitt - 14040354
Andrew Goode - 13852898
Thomas Dodgson - 13887791

University of Technology Sydney

Autumn — 2025

Contents

Project Overview: WaitFotMe

We are collaborating with two other groups to create a robot-guided art gallery experience that includes drink service. Our team's contribution focuses on TurtleBot3 robots that autonomously deliver drinks while navigating the gallery space and avoiding both known and detected obstacles. They are designed as the wait staff for the patrons of the gallery.

User Assumptions: This documentation assumes the user has some familiarity with Ubuntu and ROS 2 basics. However, all setup steps are clearly explained for new users.

Demonstration Purposes: for `Simulation` Assuming the computer is already set up please follow steps in section `5.1` to launch the simulation and `5.2` for adding goals. When demonstrating `Hardware`, the assumption is the TurtleBot has already been launched with namespace. Therefore, to run on Hardware refer to section `6.4` and `6.5` for running and adding goals respectively.

System Features and Architecture

The system has been modularised into four subsystems:

1. **Path Planning and Object Avoidance:** Calculates the shortest navigable route to target locations using A* (A-star) pathfinding logic and dynamically updates the route when new obstacles are detected.
2. **Movement Logic:** Reduces delay between movements, manages a queue of goals, and ensures indefinite operation of multiple robots.
3. **Simultaneous Operation and Goal Distribution:** Allows both robots to be managed from a single computer, and handles task assignment to optimise robot usage.
4. **Initialisation and Object Detection:** Uses visual (camera) and LiDAR data to localise each TurtleBot within the environment and to detect, classify, and publish both known and unknown obstacles in real-time.

Dependencies

3.1 Hardware

- 2–5 TurtleBot3 Waffle Pi robots
- 360°2D LiDAR sensors
- Logitech USB webcams
- Central Ubuntu 22.04 PC with ROS 2 Humble

3.2 Software

- ROS 2 Humble
- RViz 2
- OpenCV
- Gazebo Simulator (for simulation only)
- Git Repository waitforme

Software Setup

4.1 Installing Software

This section provides a guide to install the software dependencies as listed above. It is recommended that each line is run separately, to avoid errors or improper installation.

1. **Open terminal on your computer**
2. **Install ROS 2 and Dependencies**

```
sudo apt update
sudo apt install ros-humble-desktop
```

3. **Set Up Workspace and rosdep**

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws
source /opt/ros/humble/setup.bash
rosdep install --from-paths src -r -y
```

4. **Install OpenCV**

```
sudo apt install libopencv-dev
```

5. **Clone Required Packages**

```
cd ~/ros2_ws/src

# Clone TurtleBot3 Simulation repository
git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3_simulations.
    git

# Clone TurtleBot3 core packages (includes Gazebo and RViz integration)
git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3.git

# Clone custom waitforme package
git clone https://github.com/Goodbudy/waitforme.git
```

6. **Build and Source the Workspace**

```
cd ~/ros2_ws
colcon build --symlink-install
source install/setup.bash
```

4.2 Directory Structure

```
ros2_ws/
 src/
    turtlebot3_simulations/
    turtlebot3/
    waitforme/
 install/
 build/
 log/
```

Running Simulation

5.1 Launching Simulation

Open your ROS2 workspace

```
cd ~/ros2_ws
```

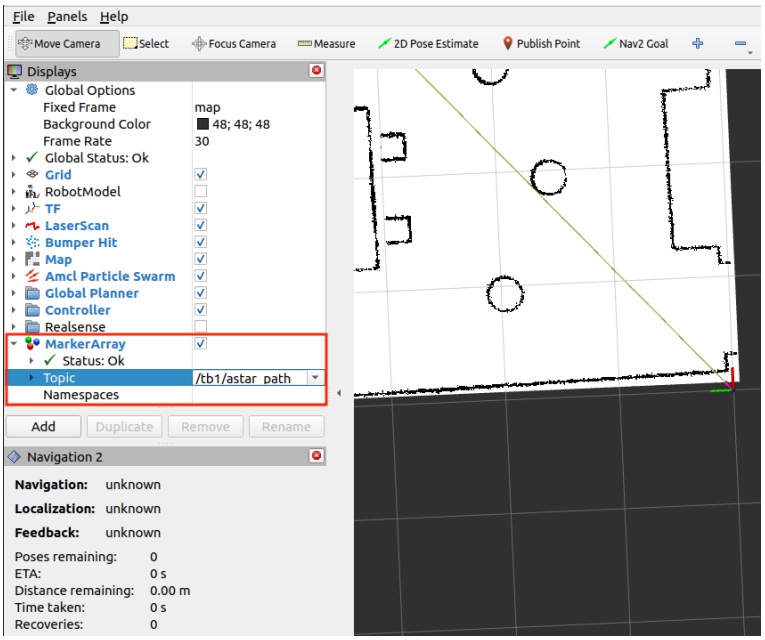
It is recommended when launching for the first time to rebuild and source your workspace

```
colcon build --symlink-install
source install/setup.bash
```

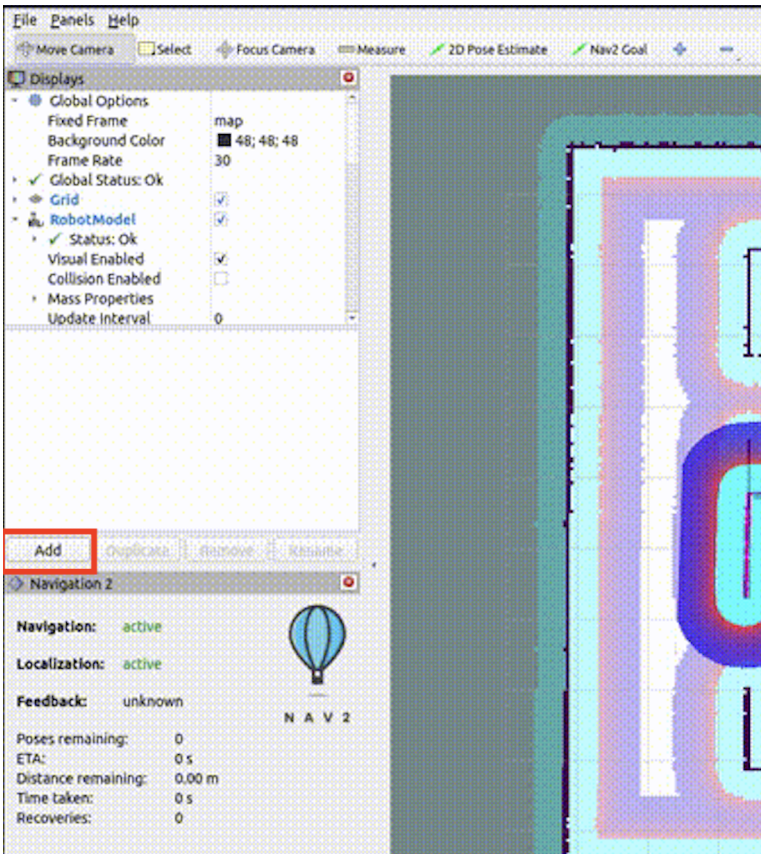
Turn off your internet and run this single command to start everything in simulation:

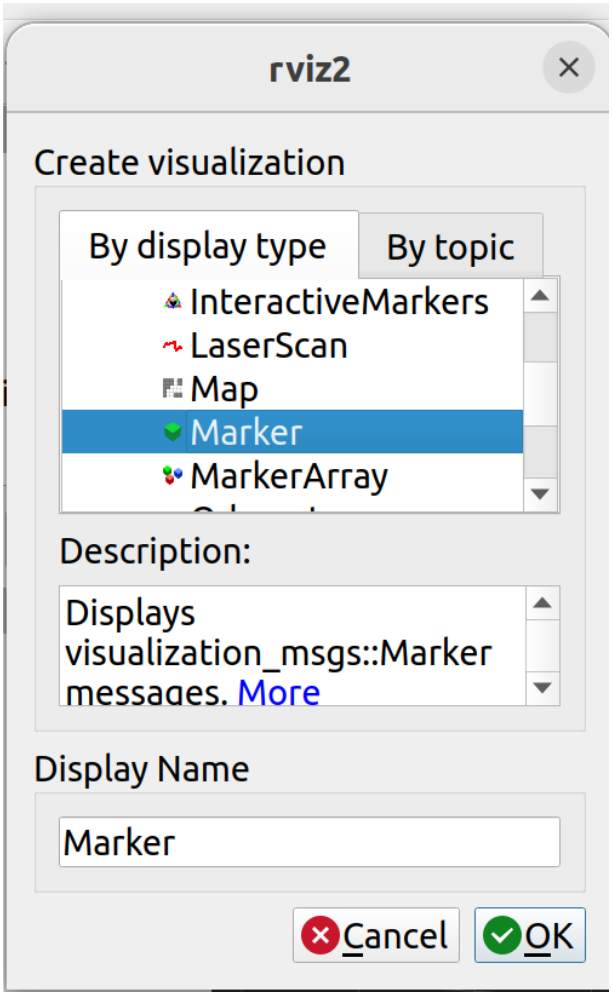
```
ros2 launch turtlebot3_multi_robot gazebo_multi_nav2_world.launch.py
```

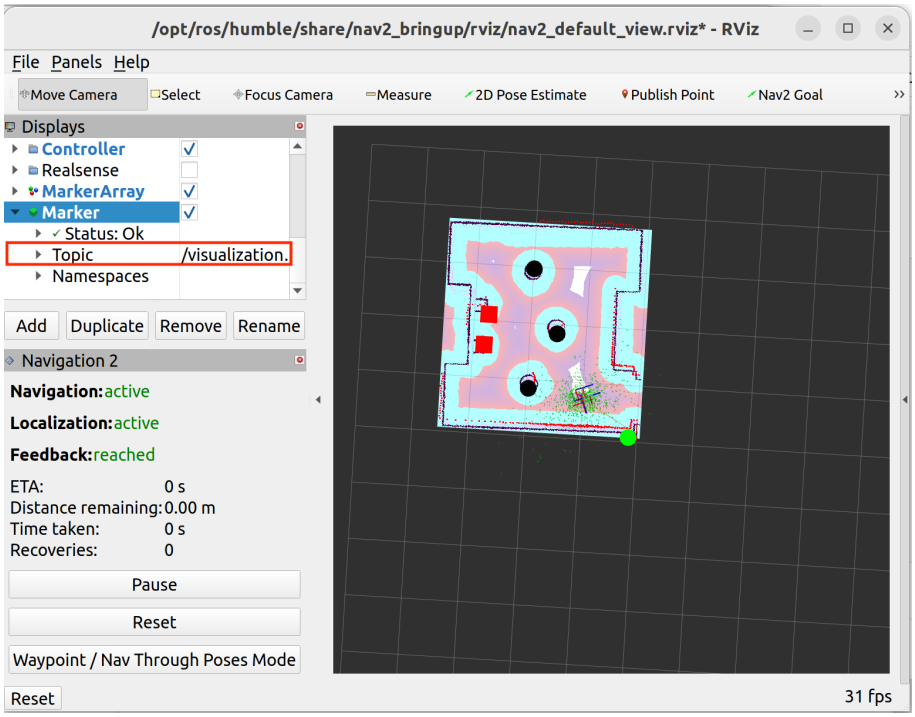
Rviz will launched after 30 seconds of simulation time. The next node will spin after another 30 seconds providing time to ensure namespaced publishers can be selected. in the first Rviz terminal change the topic in Marker Array to *tb1/astar_path* using the drop down menu.



To add obstacle detection select **Add**, scroll down to see option **Marker** and double click to select. Once selected use the drop down menu beside topic to select *tb1/visualisation_marker*.







repeat this process on the second Rviz tab using namespace tb2

5.2 Adding Goals

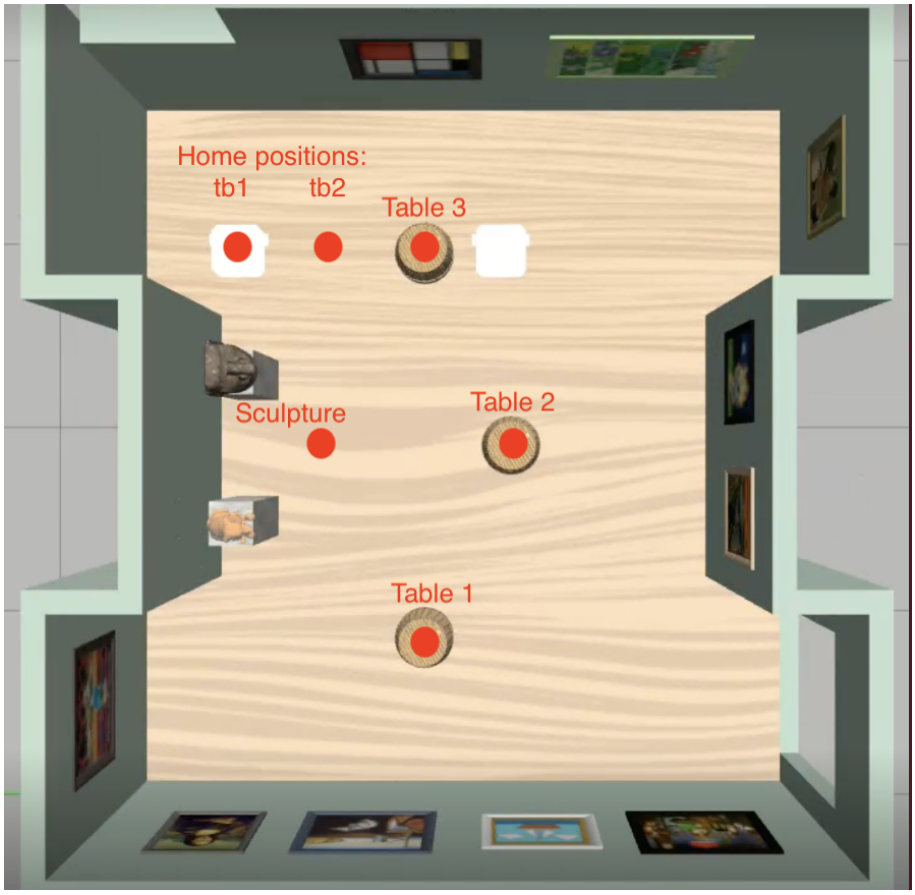
Open a new terminal window while running the simulation. Similarly navigate to the ROS2 workspace

```
cd ~/ros2_ws
```

Then add a goal using this command line:

```
add_goal table1
```

this can be used for a range of preplanned goals including, `table2 table3 sculptures tb1_home tb2_home`. You can see the associated positions in the image below:



alternatively call any position using:

```
ros2 service call /add_goal issy/srv/AddGoal "{x: 1.0, y: 0.8}"
```

the x and y values in this command can be replaced with any values as long as they are within the bounds of the map and are not at the same location as an obstacle.

Hardware Setup and Deployment

6.1 Building the Physical Map

Use foam walls, 0.2m cube blocks (sculptures), and 0.3m cylindrical tables. Ensure good lighting.

6.2 Mapping the Environment (SLAM)

Follow these terminal steps to generate the YAML and PGM files from your real-world setup. **Password: turtlebot**. It recommended you use the same Domain ID as your TurtleBot ID.

Terminal 1: (on TurtleBot):

```
ssh ubuntu@192.168.0.2XX
export ROS_DOMAIN_ID=<your_id>
ros2 launch turtlebot3_bringup robot.launch.py
```

Terminal 2: Launch world in Gazebo (if simulating)

```
export TURTLEBOT3_MODEL=waffle_pi
ros2 launch turtlebot3_gazebo Gallery_Test2.launch.py
```

Terminal 3: Start Nav2 stack

```
ros2 launch nav2_bringup navigation_launch.py use_sim_time:=True
```

Terminal 4: Start SLAM Toolbox

```
ros2 launch slam_toolbox online_async_launch.py use_sim_time:=True
```

Terminal 5: Start RViz

```
ros2 run rviz2 rviz2 -d /opt/ros/humble/share/nav2_bringup/rviz/
   nav2_default_view.rviz
```

Terminal 6: Teleoperate TurtleBot

```
export TURTLEBOT3_MODEL=waffle_pi
ros2 run turtlebot3_teleop teleop_keyboard
```

Terminal 7: Save the map

```
ros2 run nav2_map_server map_saver_cli -f my_map
```

- Once the SLAM-generated map is saved (as a .pgm and .yaml pair), it can be used in localisation and navigation tasks.
- Note: The launch files that utilise the map (i.e., for navigation post-integration) will be confirmed and finalised during the integration phase.

The generated .yaml and .pgm files should be moved to the **worlds** folder inside of the **turtlebot3_multi_robot** package. Additionally inside the **launch** folder within the same package update `source_code_multi.launch.py` to ensure the string containing the path to the map is pointing to the correct files.

```python
import os
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, TimerAction, IncludeLaunchDescription
from launch_ros.actions import Node
from launch.substitutions import LaunchConfiguration
from launch.launch_description_sources import PythonLaunchDescriptionSource
from ament_index_python.packages import get_package_share_directory

def generate_launch_description():
    ld = LaunchDescription()

    robots = [
        {'name': 'tb1'},
        {'name': 'tb2'},
    ]

    use_sim_time = LaunchConfiguration('use_sim_time', default='false')

    # Declare use_sim_time argument
    ld.add_action(DeclareLaunchArgument('use_sim_time', default_value='false'))

    # Load real map path
    real_map = os.path.join(
        get_package_share_directory('turtlebot3_multi_robot'),
        'worlds', 'real_map4.yaml')

    # Launch the GoalManager node (non-namespaced)
    ld.add_action(Node(
        package='turtlebot3_multi_robot',
        executable='manager',
        name='manager',
        output='screen'
    ))

```

6.3 Setting Up Namespaced Robots

1. Open a terminals for the main computer.
2. Open an additional termal for the TurtleBot.
3. In the terminal for the TurtleBot, SSH into the TurtleBot in their own unique terminal using:

   ```
   ssh ubuntu@<ip-address>
   ```

   The IP address for in-lab testing is 192.168.0.2XX. where XX is representative of the robots ID.

4. Create and set up the workspace on each TurtleBot:

   ```
   mkdir -p ~/rs2_ws/src
   cd ~/rs2_ws/src
   ros2 pkg create multi_robot_bringup --build-type ament_cmake
   cd multi_robot_bringup
   mkdir launch
   ```

5. Replace contents of `CMakeLists.txt` with:

   ```
   echo "cmake_minimum_required(VERSION 3.5)
   project(multi_robot_bringup)

   find_package(ament_cmake REQUIRED)

   install(DIRECTORY
   ```

```
  launch
  DESTINATION share/${PROJECT_NAME}
)

ament_package()" > CMakeLists.txt
```

6. From the terminal for the central computer, copy the launch file into the launch folder for the TurtleBot:

```
scp launch/namespaced_robot.launch.py ubuntu@<ip-address>:~/rs2_ws/src
   /multi_robot_bringup/launch/
```

7. Repeat steps 2 to 6 for each additional TurtleBot.

8. On each TurtleBot:

```
cd ~/rs2_ws
colcon build --symlink-install
source install/setup.bash
ros2 launch multi_robot_bringup namespaced_robot.launch.py namespace
   :=<robot_name>
```

   Use namespaces in the format `tb1`, `tb2`, ..., `tb5`.

9. On the central computer, run the multi-robot RViz launch file:

```
ros2 launch turtlebot3_multi_robot multi_robot_nav2_bringup.launch.py
```

10. The `general_settings.yaml` is pre set for 2 TurtleBots. If deploying more than 2 robots, update `general_settings.yaml` in `params/` to include additional robots in the following format:

```
robots:
  - name: tb1
    x_pose: 0.0
    y_pose: 0.0
    z_pose: 0.01
  - name: tb2
    x_pose: 0.0
    y_pose: 0.0
    z_pose: 0.01
  - name: tb3
    x_pose: 0.0
    y_pose: 0.0
    z_pose: 0.01
  - name: tb4
    x_pose: 0.0
    y_pose: 0.0
    z_pose: 0.01
```

```
  - name: tb5
    x_pose: 0.0
    y_pose: 0.0
    z_pose: 0.01
```

6.4 Running Hardware

Once the Turtlebots have been launched with the correct namespace, using a seperate terminal on your comupter launch:

```
ros2 launch turtlebot3_multi_robot multi_robot_nav2_bringup.launch.py
```

this will bring up Rviz Nav stack per Turtlebot.
in the LaserScan drop down menu ensure the topic `tb1/scan` includes its namespace. repeat this for additional Rviz terminals.

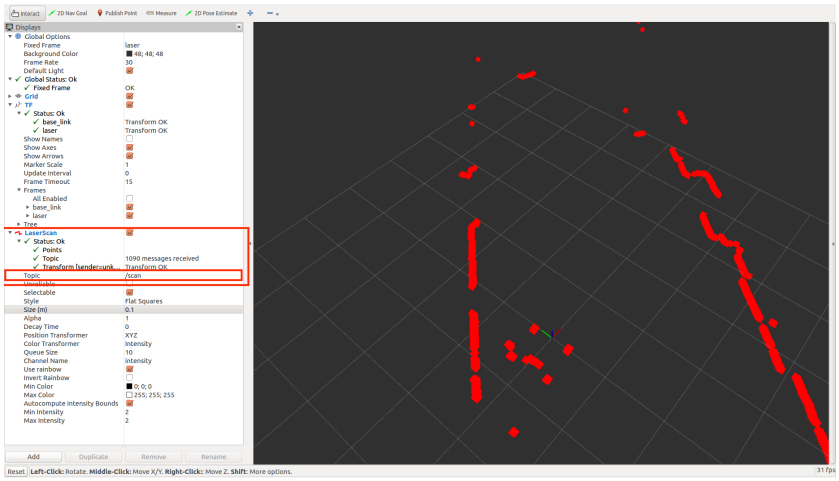
Be Patient

The system may be slow to load or your TurtleBot may not appear. Attempt to 2D pose and wait.
If you wait a minute and it still has not loaded in open a new terminal and republish the pose using this command.

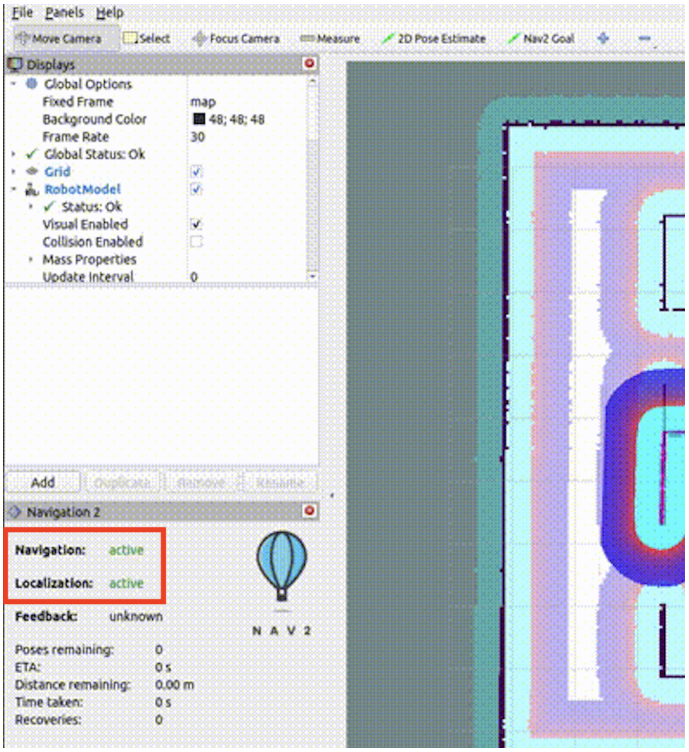
```
for i in {1..5}; do
  ros2 topic pub /tb1/initialpose geometry_msgs/PoseWithCovarianceStamped "
  header:
    frame_id: 'map'
  pose:
    pose:
      position:
        x: 1.0
        y: 1.0
        z: 0.0
      orientation:
        x: 0.0
        y: 0.0
        z: 0.0
        w: 1.0
    covariance: [0.25, 0, 0, 0, 0, 0,
                 0, 0.25, 0, 0, 0, 0,
                 0, 0, 0.0, 0, 0, 0,
                 0, 0, 0, 0.068538919, 0, 0,
                 0, 0, 0, 0, 0.068538919, 0,
                 0, 0, 0, 0, 0, 0.068538919]" -1
  sleep 1
done
```

repeat this per robot simply changing tb1/initialpose to tb2/initialpose.

By now the robot should be visible in Rviz. use 2D pose to position the robot and align the laser scans of the real environment with the map.



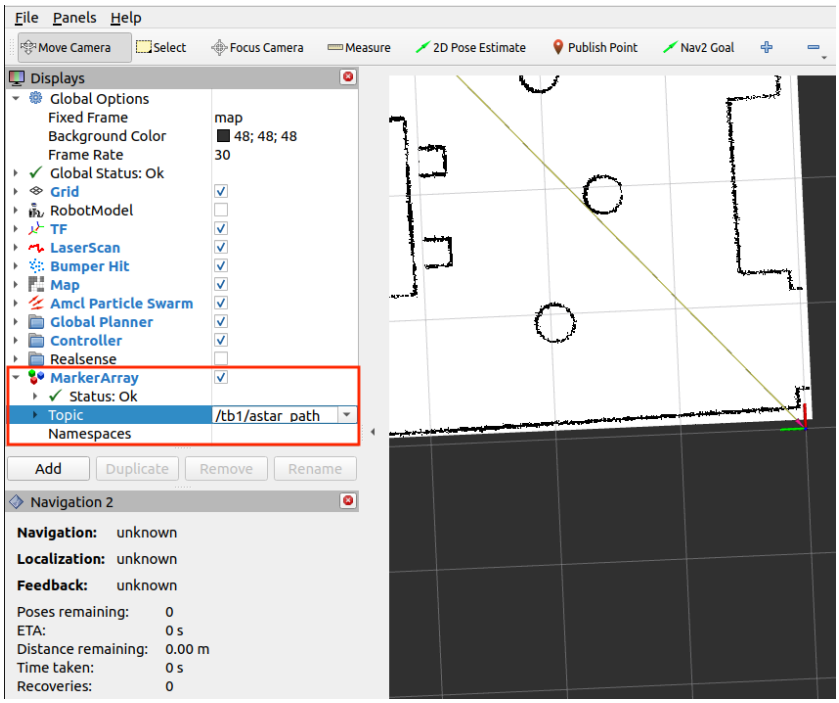
In Rviz ensure the robots navigation and localisation are `active`.
Refer to the image below.



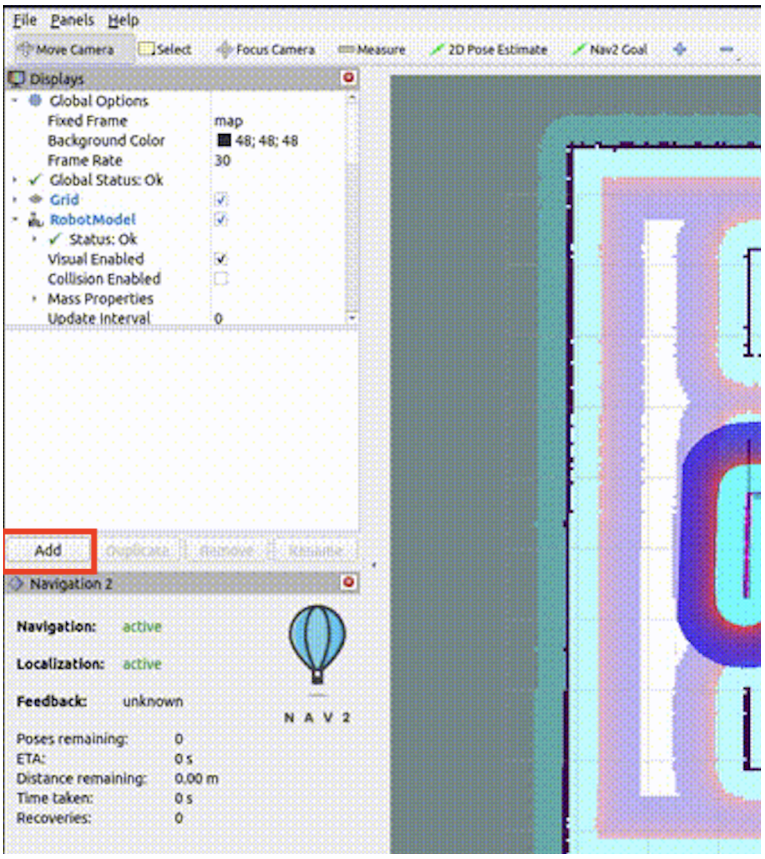
If the above conditions are true run:

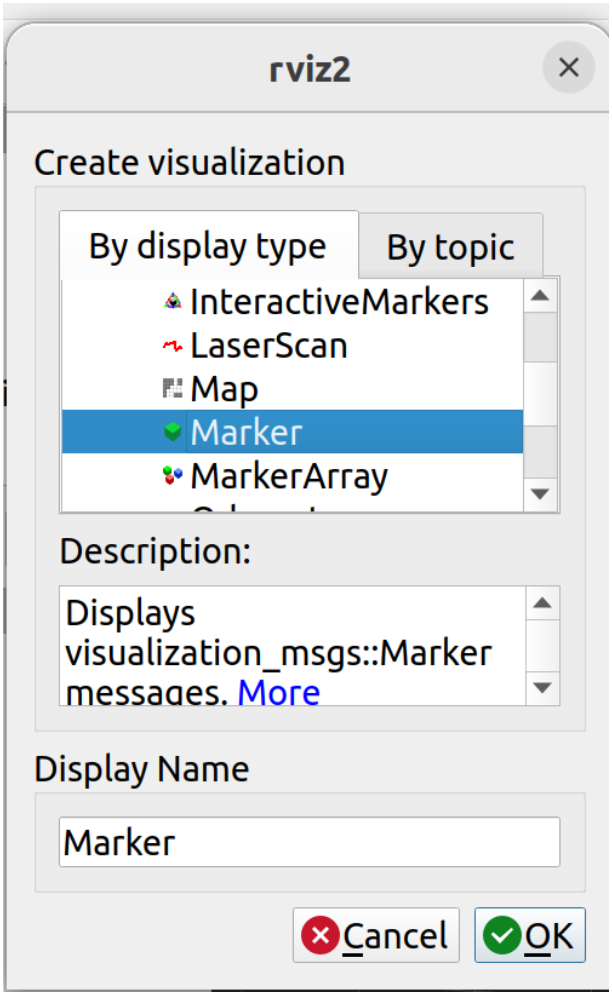
```
source_code_multi.launch.py
```

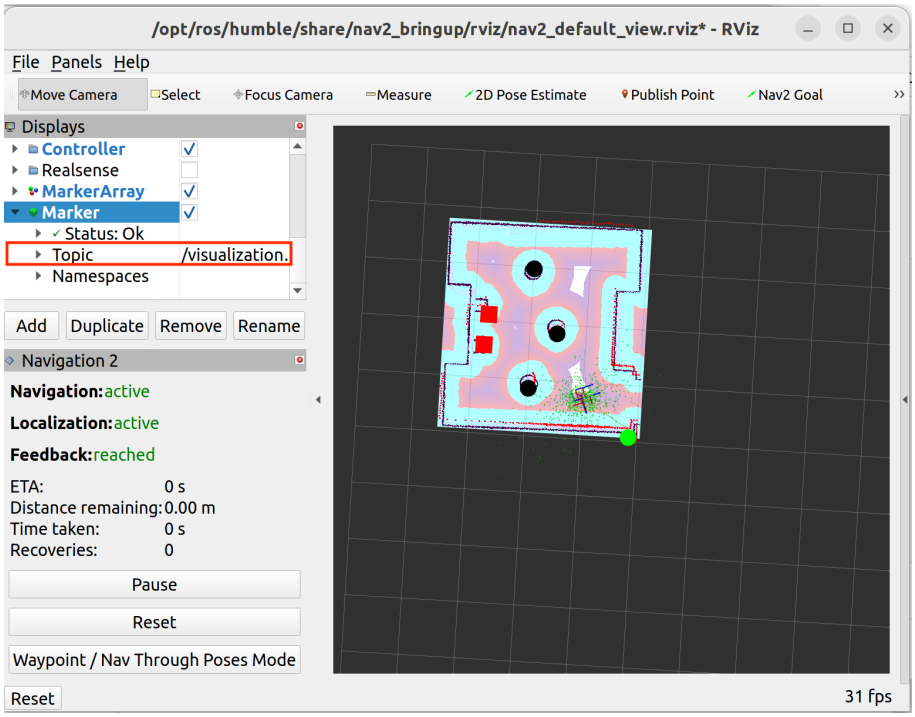
In the Rviz terminal associated with tb1 change the topic in Marker Array to *tb1/astar_path* using the drop down menu.



To add obstacle detection select **Add**, scroll down to see option **Marker** and double click to select. Once selected use the drop down menu beside topic to select tb1/visualisation_marker.







repeat this process on the second Rviz tab using namespace tb2

6.5 Adding Goals

Open a new terminal window while running the simulation. Similarly navigate to the ROS2 workspace

```
cd ~/ros2_ws
```

Then add a goal using this command line:

```
add_goal table1
add_goal table2
```

Alternatively you can choose any point on the map using:

```
ros2 service call /add_goal issy/srv/AddGoal "{x: 0.0, y: 1.0}"
```

the x and y values in this command can be replaced with any values as long as they are within the bounds of the map and are not at the same location as an obstacle.

6.6 Camera Operation

Open a two new terminal window and run the following command in both.

```
cd ~/ros2_ws
```

Then in the first terminal open the image viewer using

```
ros2 run rqt_image_view rqt_image_view
```

In the second terminal run the camera node

```
ros2 run tom people_detector
```

The topic the image viewer is subscribed to may need to be changed to be */people_detector/image*. The computer should then use the inbuilt webcam and display an image of what the webcam sees. When placed far enough away to capture an entire person, the image will display a green rectangle around the person while they remain in view.

Expected Outcomes

7.1 Multi-Turtlebot Simulation

- Black circles and red squares represent objects in RViz
- Robots follow A* path and PNG path visualisation saved
- Two or more robots navigate and act simultaneously

7.2 Single TurtleBot Hardware

- Black circles and red squares represent objects in RViz
- Robots follow A* path and PNG path visualisation saved
- One robot navigates to and from goals

Known Issues and Workarounds

- **Robot Always Busy:** If the robot busy status never changes to false, the notify IDLE service was not registered during its 1 second ping. please restart the launch process to reset the service. Alternatively you can run:

  ```
  notify_idle tb1
  notify_idle tb2
  ```

  this will cause the robot to receive the next goal but will override the movement node requiring the system to be relaunched anyway.

- **Robot Unresponsive:** Ensure initial pose is set. You can attempt to republish pose by using 2D pose or using this command in terminal with the appropriate namespace:

  ```
  for i in {1..5}; do
    ros2 topic pub /tb1/initialpose geometry_msgs/
       PoseWithCovarianceStamped "
    header:
      frame_id: 'map'
    pose:
      pose:
        position:
          x: 1.0
          y: 1.0
          z: 0.0
        orientation:
          x: 0.0
          y: 0.0
          z: 0.0
          w: 1.0
      covariance: [0.25, 0, 0, 0, 0, 0,
                   0, 0.25, 0, 0, 0, 0,
                   0, 0, 0.0, 0, 0, 0,
                   0, 0, 0, 0.068538919, 0, 0,
                   0, 0, 0, 0, 0.068538919, 0,
                   0, 0, 0, 0, 0, 0.068538919]" -1
    sleep 1
  done
  ```

- **No Markers or Paths Displaying on Map:** Ensure the Rviz topics such as `scan` are namespaced and that the data for `Marker Array` and `Marker` are subscribing to topics *ns/astar_path* and *ns/visualisation_marker*.

- **No TF Data:** Reboot machine and check namespacing and time sync.

- **Gazebo Won't Load:** Ensure wi-fi is off. try again. Reboot machine.

- **Gazebo Won't Open:** Reboot machine.

- **Services Unavailable:** Reboot machine.

Team Contributions and Contact

- Path Planning — Andrew Goode
- Movement Logic — Issy Pitt
- Goal Management — Hallie Robins
- SLAM & Object Detection — Thomas Dodgson

Contact: `hallie.r.robins@student.uts.edu.au`