Water Turret AGV (Automated Guided Vehicle)

A PROJECT REPORT

Submitted by

BUVICA M [1RVU23CSE123]

GOPIKA R [1RVU23CSE170]

MOHAN CHANDRA SS [1RVU23CSE282]

NEERAJ P RAO [1RVU23CSE303]

as part of the ***Semester End Examination***

for the course

CS2112 Introduction to ROS

Offered in

Semester IV

of

B.Tech (Hons)

School of Computer Science and Engineering

RV University

RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

MAY & 2025

DECLARATION

I, **Buvica M (1RVU23CSE123), Gopika R(1RVU23CSE170), Mohan Chandra SS (1RVU23CSE282), Neeraj P Rao (1RVU23CSE303) ,** student Fourth B.Tech in **Computer Science & Engineering,** at School of Computer Science and Engineering, **RV University,** hereby declare that the project work titled **"Water Turret AGV (Automated Guided Vehicle)"** has been carried out by us and submitted in partial fulfilment for the **Semester End Examination** for the course **CS2112 Introduction to ROS** during the academic year **2023-2024**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Buvica M (1RVU23CSE123)	Signature
Gopika R (1RVU23CSE170)	Signature
Mohan Chandra SS (1RVU23CSE282)	Signature
Neeraj P Rao (1RVU23CSE303)	Signature

Place: RV University

Date: 28/05/2025

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

School of Computer Science and Engineering

RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India – 562112

CERTIFICATE

This is to certify that the project work titled **“Water Turret AGV (Automated Guided Vehicle)"** is performed by **Buvica M (1RVU23CSE123),Gopika R(1RVU23CSE170),Mohan Chandra SS (1RVU23CSE282), Neeraj P Rao (1RVU23CSE303),** a bonafide students of Bachelor of Technology at the School of Computer Science and Engineering, RV university, Bengaluru as part of the **Semester End Examination** for the course **CS2112 Introduction to ROS** offered in Semester IV of Bachelor of Technology in Computer Science & Engineering , during the Academic year **2024-2025**.

Course Faculty	Program Head(Minor and Electives)
SOCSE	SOCSE
RV University	RV University
Date:	Date:

Name of the Examiner	Signature of Examiner
1.	
2.	

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to **RV University (RVU)**, the **School of Computer Science and Engineering (SOCSE)**, the **Dean**, and our **Program Head, Ms. Merin Thomas**, for their unwavering support and guidance throughout this journey.

We sincerely thank **Prof. Thotreithem Hongray** for his invaluable knowledge, encouragement, and mentorship.

Our deepest appreciation goes to our friends for their constant motivation and camaraderie.

We are forever indebted to our parents and family for their unconditional love, sacrifices, and belief in us. Their encouragement has been our greatest strength.

This accomplishment would not have been possible without the collective support of everyone who stood by us.

Thank you all for being an integral part of our journey.

Date: 28-05-25

BUVICA M [1RVU23CSE123]

GOPIKA R [1RVU23CSE170]

MOHAN CHANDRA SS [1RVU23CSE282]

NEERAJ P RAO [1RVU23CSE303]

Place: RV university, Bengaluru

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

1.1 Introduction

In industrial settings such as warehouses and factories, robotic automation plays an increasingly important role in improving operational safety, monitoring, and efficiency. Traditional fire response mechanisms, including fixed sprinklers and manual fire fighting systems, are often limited in coverage and can expose human personnel to hazardous conditions. As industrial environments become more complex and dynamic, mobile robotic systems offer a promising solution for enhancing responsiveness and flexibility.

This project presents a simulated Autonomous Ground Vehicle (AGV) developed using ROS. The AGV is equipped with a differential drive base for movement, standard RGB cameras for visual monitoring, and a LIDAR sensor for obstacle detection and navigation. A turret mechanism is included in the robot model to represent future actuation capabilities, although it is currently static and non-functional. The primary focus of this phase is on simulating the robot's movement and sensor integration within a virtual environment using Gazebo, and visualizing the system's operation using RViz. While fire detection and active turret control are planned for future development, this project establishes the foundational architecture for such enhancements.

1.2 Motivation of the Design

The design of this simulated AGV stems from the need to create a flexible, mobile robotic platform that can navigate industrial environments and lay the groundwork for future safety-related functionalities. In large-scale facilities like warehouses or factories, traditional safety systems are often fixed and inflexible, making them less effective in responding to dynamic or localized hazards.

By simulating an AGV equipped with essential mobility and perception components, this project aims to build a modular base that can later be extended with more advanced features such as turret actuation and fire detection. The use of ROS 2 enables a structured and scalable software architecture, making it easier to integrate new hardware, sensors, or decision-making algorithms over time.

The inclusion of standard RGB cameras and LIDAR allows the robot to perceive its surroundings and navigate obstacles in a simulated environment. Although the turret is currently static, its design in the URDF/Xacro files ensures that future development can activate and control it through ROS interfaces. This incremental and modular approach reflects a realistic path toward building autonomous, responsive systems suitable for real-world industrial applications.

2.0 THE MODEL

This section outlines the structural and sensory components of the Water Turret AGV, detailing its robot description files (URDF/XACRO), Gazebo integration, sensors, and ROS 2-based communication framework.

Base and Mobility

- **Base Link**: Central structural body of the robot, serving as the anchor for all other components.

- **Left and Right Wheel Links**: Attached to the base via **continuous joints**, enabling differential drive mobility.

- **Caster Wheel (optional)**: Mounted at the rear or front to stabilize the base; typically fixed or passively rotating.

Turret System

- **Turret Link**: Mounted on top of the base with a **revolute joint**, allowing horizontal rotation for targeting.

- The turret serves as the platform for the arm and nozzle, enabling directed spraying in future implementations.

Arm and Nozzle

- **Arm Link 1**: Connected to the turret via a **revolute joint** to enable vertical articulation.

- **Arm Link 2**: Attached to Arm 1 with a second **revolute joint**, extending the range and flexibility.

- **Nozzle Link**: Terminal link of the arm system; currently static but positioned for simulated fire-suppression.

Sensor Assembly

- **Camera Links**: Multiple camera links are attached to the base and turret (e.g., `camera_link_front`, `camera_link_turret`) to provide comprehensive RGB vision coverage.

- **LIDAR Link**: Mounted on top or front of the base; simulated using a ray sensor plugin to provide 2D spatial awareness.

- All sensor links are **fixed joints**, ensuring stable placement during movement.

Kinematic Structure

- All links are connected hierarchically through joints, forming a complete **kinematic tree** rooted at the base link.

- Each joint is defined with appropriate motion limits, effort constraints, and damping values to ensure realistic simulation behavior in Gazebo.

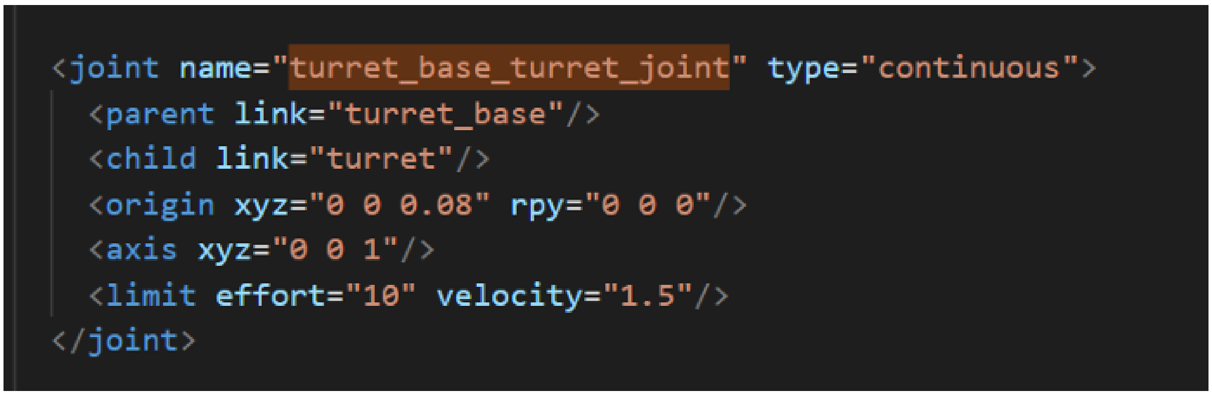
Transform Publishing

- The robot's joint states are broadcast through /joint_states, and the complete link tree is visualized via /robot_state_publisher, allowing real-time observation in **RViz**.

2.1 Various Elements of the Model

2.1.1 The URDF Descriptions

The robot model is structured using the Unified Robot Description Format (URDF), authored in modular XACRO files for better readability and parameter control. The base of the AGV is defined as a rectangular chassis (**base_link**) connected to four wheels via continuous joints. The model also includes a turret and nozzle assembly with articulated joints.



```
<joint name="turret_base_turret_joint" type="continuous">
  <parent link="turret_base"/>
  <child link="turret"/>
  <origin xyz="0 0 0.08" rpy="0 0 0"/>
  <axis xyz="0 0 1"/>
  <limit effort="10" velocity="1.5"/>
</joint>
```

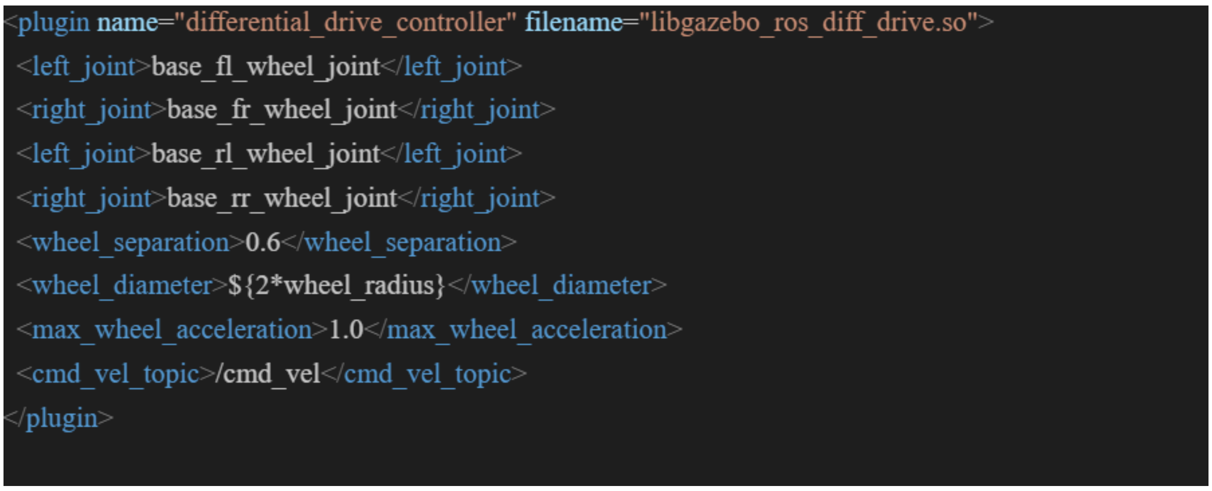
Fig. 2.1.1 – Continuous yaw joint definition for turret in XACRO

2.1.2 XACRO and Gazebo Descriptions

The robot's full structure is defined in my_dd_robot_model.xacro, which includes parameterized macros for link dimensions, mass, visuals, and collision properties. Gazebo-specific elements are included via .gazebo tags to link simulation plugins.

- The differential drive is implemented using the libgazebo_ros_diff_drive.so plugin, connected to four wheel joints for base motion.

- Gazebo plugins are also used to simulate camera and LIDAR behavior.

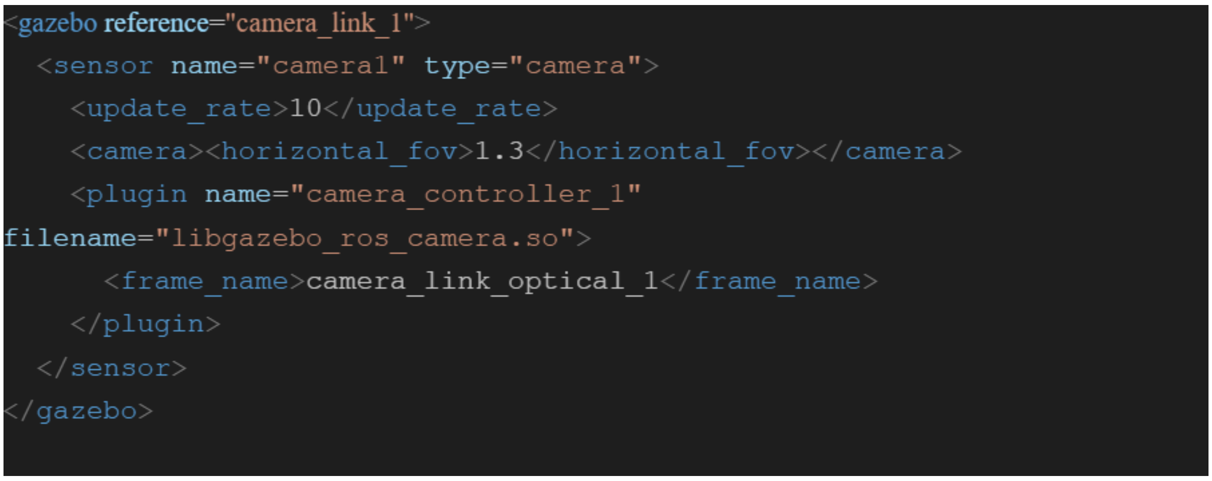


```
<plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">
 <left_joint>base_fl_wheel_joint</left_joint>
 <right_joint>base_fr_wheel_joint</right_joint>
 <left_joint>base_rl_wheel_joint</left_joint>
 <right_joint>base_rr_wheel_joint</right_joint>
 <wheel_separation>0.6</wheel_separation>
 <wheel_diameter>${2*wheel_radius}</wheel_diameter>
 <max_wheel_acceleration>1.0</max_wheel_acceleration>
 <cmd_vel_topic>/cmd_vel</cmd_vel_topic>
</plugin>
```

Fig. 2.1.2 – Differential drive plugin handling /cmd_vel commands

2.1.3 Sensors – Camera and LIDAR

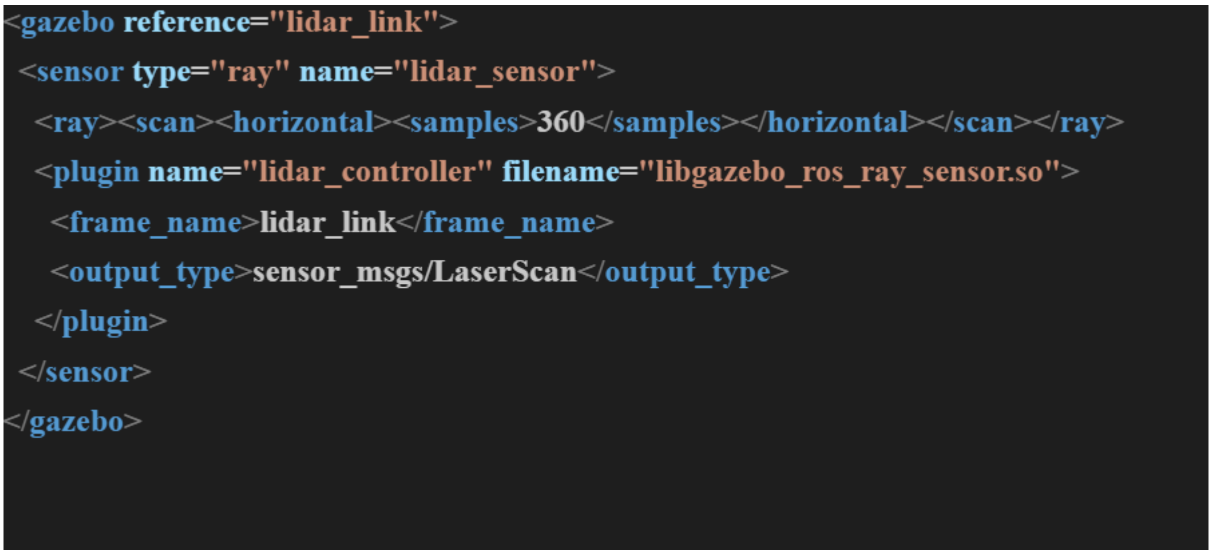
The AGV includes five RGB cameras for environmental perception, mounted at the front, back, left, right, and on the turret. These cameras are defined in XACRO and integrated into the Gazebo simulation using the gazebo_ros_camera plugin.



```
<gazebo reference="camera_link_1">
  <sensor name="camera1" type="camera">
    <update_rate>10</update_rate>
    <camera><horizontal_fov>1.3</horizontal_fov></camera>
    <plugin name="camera_controller_1"
filename="libgazebo_ros_camera.so">
      <frame_name>camera_link_optical_1</frame_name>
    </plugin>
  </sensor>
</gazebo>
```

Fig. 2.1.3 – Camera plugin integration in Gazebo for camera_link_1

The LIDAR sensor is placed on top of the robot and simulates a 2D scan with 360° coverage using the gazebo_ros_ray_sensor plugin.



```
<gazebo reference="lidar_link">
 <sensor type="ray" name="lidar_sensor">
  <ray><scan><horizontal><samples>360</samples></horizontal></scan></ray>
  <plugin name="lidar_controller" filename="libgazebo_ros_ray_sensor.so">
   <frame_name>lidar_link</frame_name>
   <output_type>sensor_msgs/LaserScan</output_type>
  </plugin>
 </sensor>
</gazebo>
```

Fig. 2.1.4 – LIDAR sensor plugin for publishing /scan data

2.2 Details of the Nodes

2.2.1 State Publisher Nodes

ROS 2 nodes are used to broadcast and synchronize the robot’s joint states and frames.

- robot_state_publisher:
  - Publishes:
    - **/tf_static and /tf**: for fixed and dynamic frame transforms
    - **/robot_description**: URDF model
  - Subscribes:
    - **/joint_states**: for real-time joint feedback
    - **/clock**: from Gazebo
- joint_state_publisher:
  - Publishes:
    - **/joint_states**: real-time joint positions
  - Used in launch files and simulation setup.

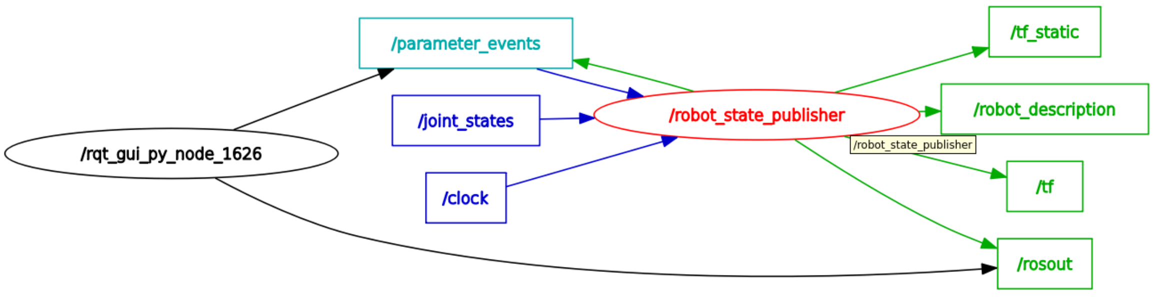


Fig:2.2.1 State of Publisher nodes

avoid.py (Obstacle Avoidance Node)

- A custom ROS 2 Python node named /obstacle_avoider
- Subscribes to /scan (LaserScan) for obstacle detection
- Publishes to /cmd_vel (Twist) to adjust robot velocity
- Implements reactive avoidance logic:
    - Stops and turns when obstacles are detected
    - Moves forward when path is clear

2.2.2 Task Details and Statements

- differential_drive_controller:
    - Subscribes to /cmd_vel for velocity commands.
    - Publishes wheel joint states to /joint_states.
- Camera Nodes:
    - Each RGB camera publishes image data on topics like /camera/front, /camera/back, etc.
    - Implemented using the gazebo_ros_camera plugin.
- LIDAR Node:
    - Publishes **/scan** with range and angle data using the gazebo_ros_ray_sensor plugin.
- Timing and Synchronization:
    - Gazebo publishes the simulation clock on /clock.
    - All nodes synchronize to this simulated time source.
- Data Flow Summary:
    - Sensors (cameras and LIDAR) publish to ROS topics.
    - Control nodes subscribe to these topics to guide robot behavior.
    - ROS 2 plugins bridge simulation physics with command/control interfaces.

3.0 IMPLEMENTATION

This section outlines the robot modeling approach, simulation environment, and ROS 2 control interfaces used to implement and test the AGV system.

3.1 Introduction to Robot Task Executed

The primary task executed by the robot involves simulating autonomous navigation and structural actuation components within a virtual industrial environment. The robot is defined using a modular and parametric **XACRO model (my_dd_robot_model.xacro)**, which enables dynamic configuration of dimensions, inertial properties, and sensor placements.

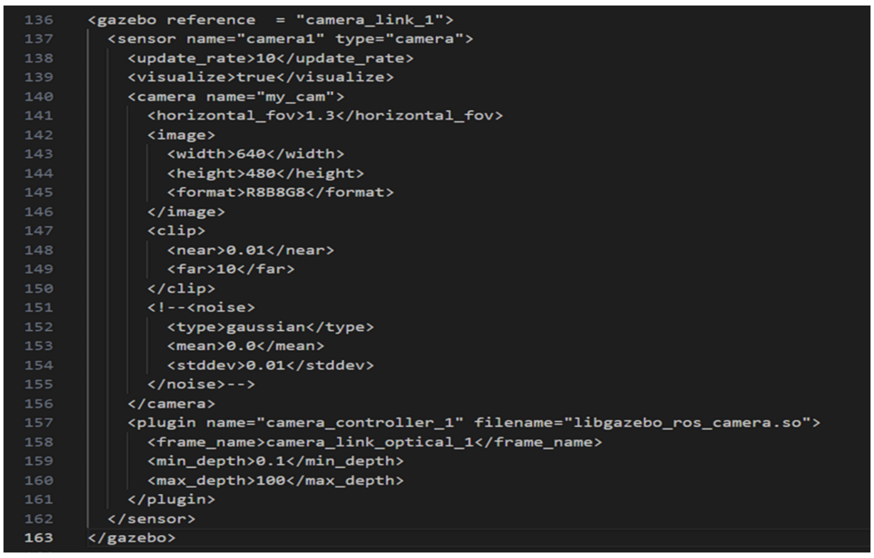
XACRO Design Highlights:

- Parametric Macros: Reusable macros simplify model maintenance by generalizing common components like inertial elements and visual properties.
- Link and Joint Definitions:

    - Chassis and Wheels: **The robot base includes base_link, four wheel links (front_left_wheel, front_right_wheel, rear_left_wheel, rear_right_wheel)** connected via continuous joints to allow differential drive mobility.

    - Turret Assembly: **A continuous yaw joint (turret_base_turret_joint)** and a **revolute pitch joint (arm_link_1_joint)** define the static turret's articulation structure.

    - Sensors: **Five standard RGB cameras are mounted at key locations (front, back, left, right, and turret)** for environment monitoring. A **lidar_link** is mounted on top of the chassis for obstacle detection.



```
<gazebo reference  = "camera_link_1">
  <sensor name="camera1" type="camera">
    <update_rate>10</update_rate>
    <visualize>true</visualize>
    <camera name="my_cam">
      <horizontal_fov>1.3</horizontal_fov>
      <image>
        <width>640</width>
        <height>480</height>
        <format>R8B8G8</format>
      </image>
      <clip>
        <near>0.01</near>
        <far>10</far>
      </clip>
      <!--<noise>
        <type>gaussian</type>
        <mean>0.0</mean>
        <stddev>0.01</stddev>
      </noise>-->
    </camera>
    <plugin name="camera_controller_1" filename="libgazebo_ros_camera.so">
      <frame_name>camera_link_optical_1</frame_name>
      <min_depth>0.1</min_depth>
      <max_depth>100</max_depth>
    </plugin>
  </sensor>
</gazebo>
```

Fig. 3.1.1 – Gazebo camera plugin configuration for RGB sensor

3.2 Gazebo Simulation

The robot is deployed in a custom Gazebo environment using the model **file my_dd_robot.gazebo.** This environment supports full physics-based simulation and integrates closely with ROS 2 for sensor data and control.

Sensor and Control Plugin Integration:

- Cameras: Each camera uses the **gazebo_ros_camera** plugin, enabling image stream publishing from different angles. These camera feeds are critical for future perception-based expansions.

- LIDAR: A 360° scanning LIDAR is implemented using the **gazebo_ros_ray_sensor plugin at the lidar_link, p**roviding real-time scan data for obstacle mapping and avoidance.

- Differential Drive: The gazebo_ros_diff_drive plugin interprets velocity commands from the /cmd_vel topic to control wheel speeds and robot motion.

ROS 2 Control and Plugin Interfaces:

- **gazebo_ros_control:** This plugin connects Gazebo's simulation elements with ROS 2 control interfaces, allowing joint-level control.

- differential_drive_controller:

    - Subscribes **to /cmd_vel** for movement instructions.

    - Publishes wheel joint states on **/joint_states.**

- TF Tree Management:

    - robot_state_publisher and joint_state_publisher manage the robot's full frame hierarchy, ensuring correct spatial transformations among base, sensors, and turret joints.

Visual Enhancements:

- The robot's visual elements are defined with materials and colors in the XACRO file, improving model clarity and aiding debugging during simulation.

Simulation Environment

The system operates within a Gazebo simulation world specifically designed to evaluate its mobility, perception, and actuation systems.

Obstacle-Rich Layout:
The environment contains static obstacles such as walls, boxes, and cylindrical pillars arranged in narrow corridors and open areas. These are strategically placed to:

- Test path planning and collision avoidance.
- Evaluate camera and LIDAR visibility, occlusion handling, and object detection.
- Challenge the robot's turret and arm articulation in confined spaces.

Sensor Simulation:

The robot is equipped with simulated sensors, including:

- A RGB camera, used for visual feedback and object recognition.
- A 2D LIDAR sensor, providing distance measurements for mapping and navigation.
- Sensor plugins simulate realistic noise, latency, and range limitations.

Physics Plugins:

Gazebo's physics engine (ODE or Bullet) is used with:

- Accurate simulation of differential drive kinematics.
- Dynamics for revolute joints in the turret and arms, with damping and effort constraints.
- Gravitational and inertial effects on all movable links.

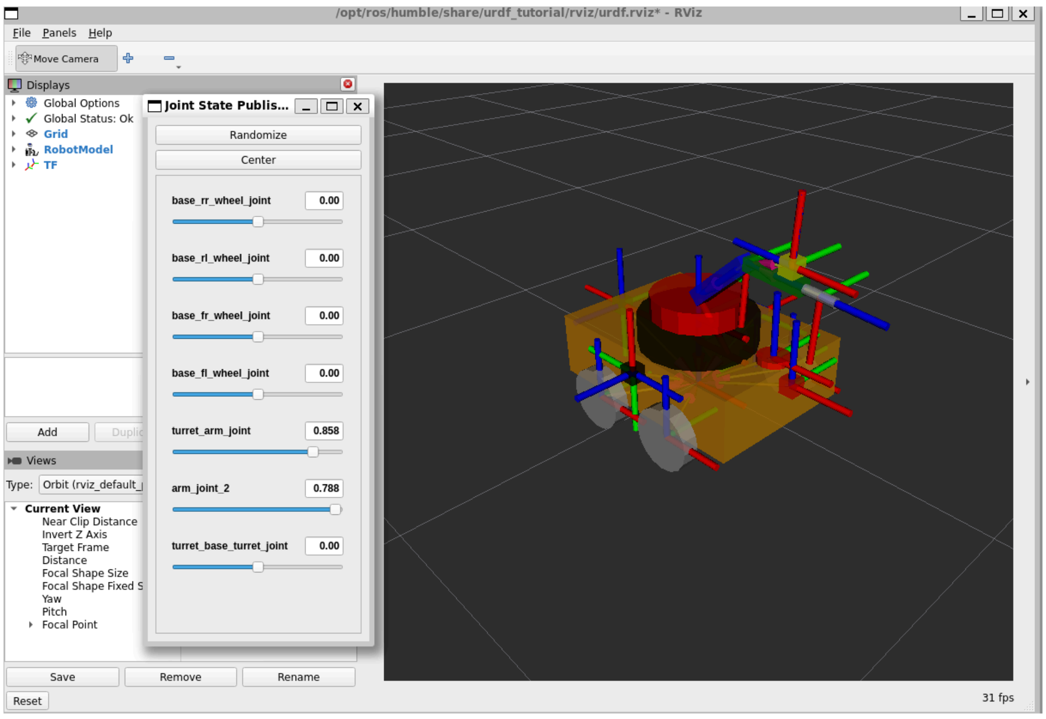
Control and Interaction Plugins:
The robot is controlled through ROS 2 interfaces, utilizing:

- Joint controllers for turret and arms.
- Diff-drive plugin for base motion.
- Teleoperation and autonomous navigation support for complete control testing.

4.0 RESULTS AND DISCUSSION

4.1 Robot Initialization and Description

The robot model, defined using XACRO and loaded into a Gazebo environment, spawns successfully with all major components, including the base, wheels, turret, cameras, and LIDAR. The **/robot_description** parameter is properly published, enabling visualization in RViz. All links and joints are correctly configured and rendered in the simulation.



Type: Orbit (rviz_default_

base_rr_wheel_joint	0.00
base_rl_wheel_joint	0.00
base_fr_wheel_joint	0.00
base_fl_wheel_joint	0.00
turret_arm_joint	0.858
arm_joint_2	0.788
turret_base_turret_joint	0.00

4.1.1 Robot Initialisation and Description

4.2 Navigation Performance and motion control

The differential drive base operates smoothly under ROS 2 control. Commands sent to the **/cmd_vel** topic enable accurate forward, reverse, and turning motions. The robot navigates through an obstacle-filled environment (boxes, walls) with consistent motion, demonstrating basic maneuverability in a simulated industrial layout.

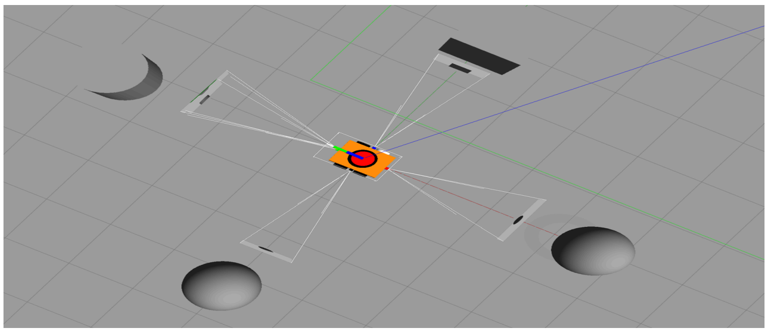


Fig 4.2.2 Navigation Performance and motion control

4.3 Sensor Feedback and Visualization

Cameras: All five RGB cameras (front, back, left, right, turret-mounted) publish image streams successfully. These feeds are visualized in both RViz and Gazebo for situational awareness and perception testing.

LIDAR: The 360° LIDAR mounted on the chassis publishes scan data to **/scan,** enabling obstacle mapping. LIDAR feedback is clearly visible in RViz, aiding navigation.

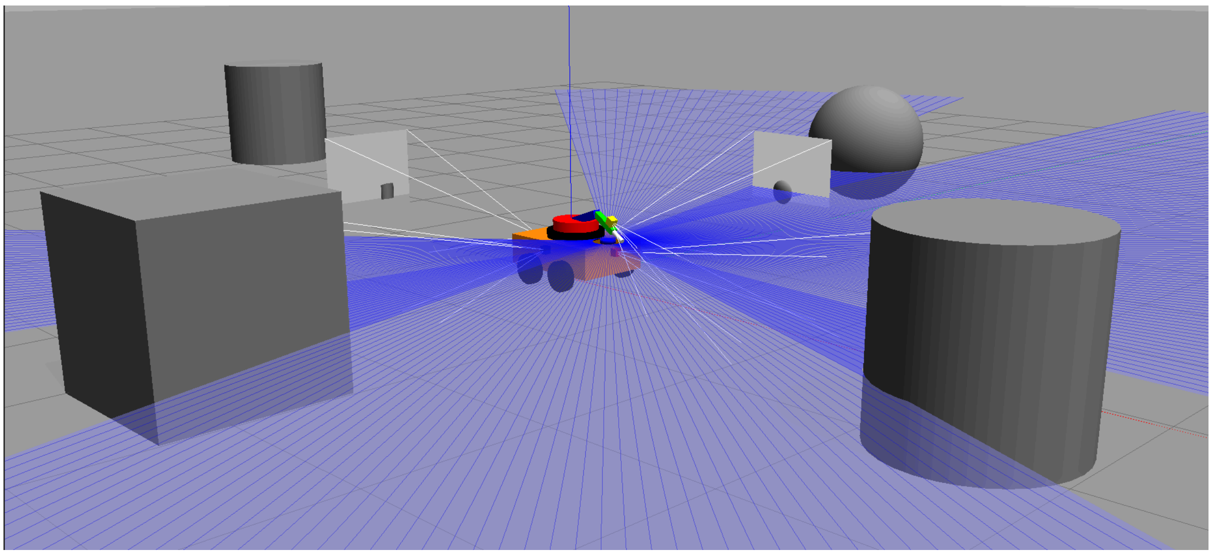
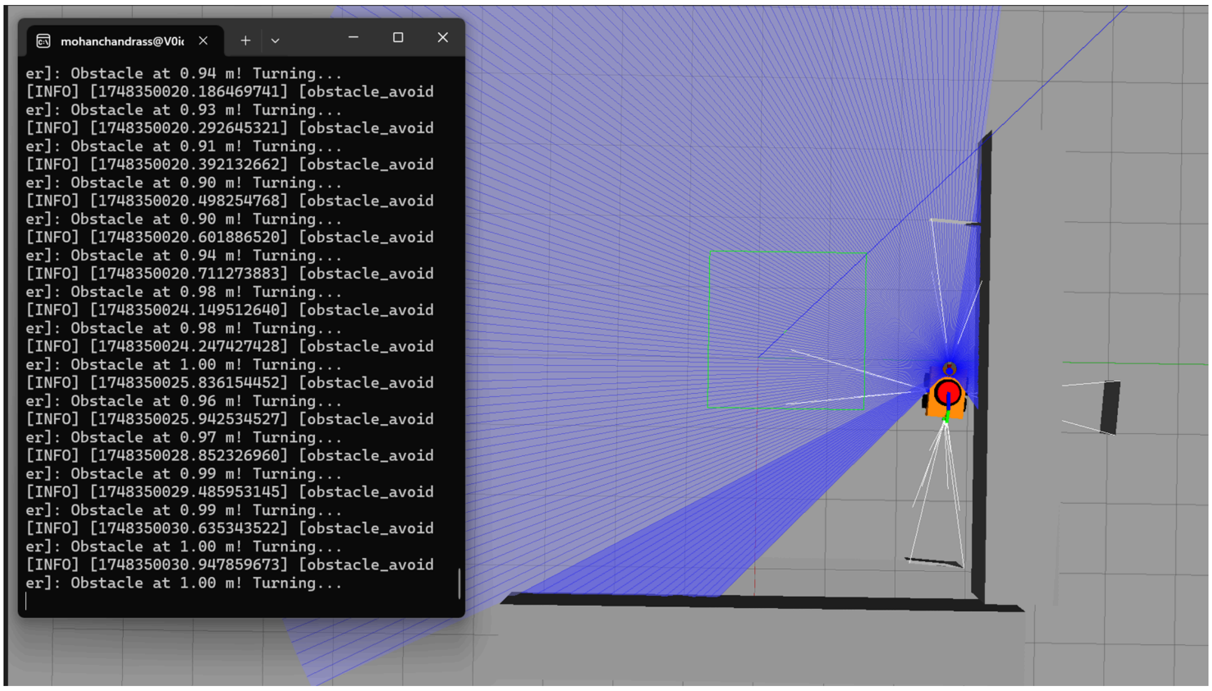


Fig 4.3.2 Sensor feedback and visualisations(1)



```
er]: Obstacle at 0.94 m! Turning...
[INFO] [1748350020.186469741] [obstacle_avoid
er]: Obstacle at 0.93 m! Turning...
[INFO] [1748350020.292645321] [obstacle_avoid
er]: Obstacle at 0.91 m! Turning...
[INFO] [1748350020.392132662] [obstacle_avoid
er]: Obstacle at 0.90 m! Turning...
[INFO] [1748350020.498254768] [obstacle_avoid
er]: Obstacle at 0.90 m! Turning...
[INFO] [1748350020.601886520] [obstacle_avoid
er]: Obstacle at 0.94 m! Turning...
[INFO] [1748350020.711273883] [obstacle_avoid
er]: Obstacle at 0.98 m! Turning...
[INFO] [1748350024.149512640] [obstacle_avoid
er]: Obstacle at 0.98 m! Turning...
[INFO] [1748350024.247427428] [obstacle_avoid
er]: Obstacle at 1.00 m! Turning...
[INFO] [1748350025.836154452] [obstacle_avoid
er]: Obstacle at 0.96 m! Turning...
[INFO] [1748350025.942534527] [obstacle_avoid
er]: Obstacle at 0.97 m! Turning...
[INFO] [1748350028.852326960] [obstacle_avoid
er]: Obstacle at 0.99 m! Turning...
[INFO] [1748350029.485953145] [obstacle_avoid
er]: Obstacle at 0.99 m! Turning...
[INFO] [1748350030.635343522] [obstacle_avoid
er]: Obstacle at 1.00 m! Turning...
[INFO] [1748350030.947859673] [obstacle_avoid
er]: Obstacle at 1.00 m! Turning...
```

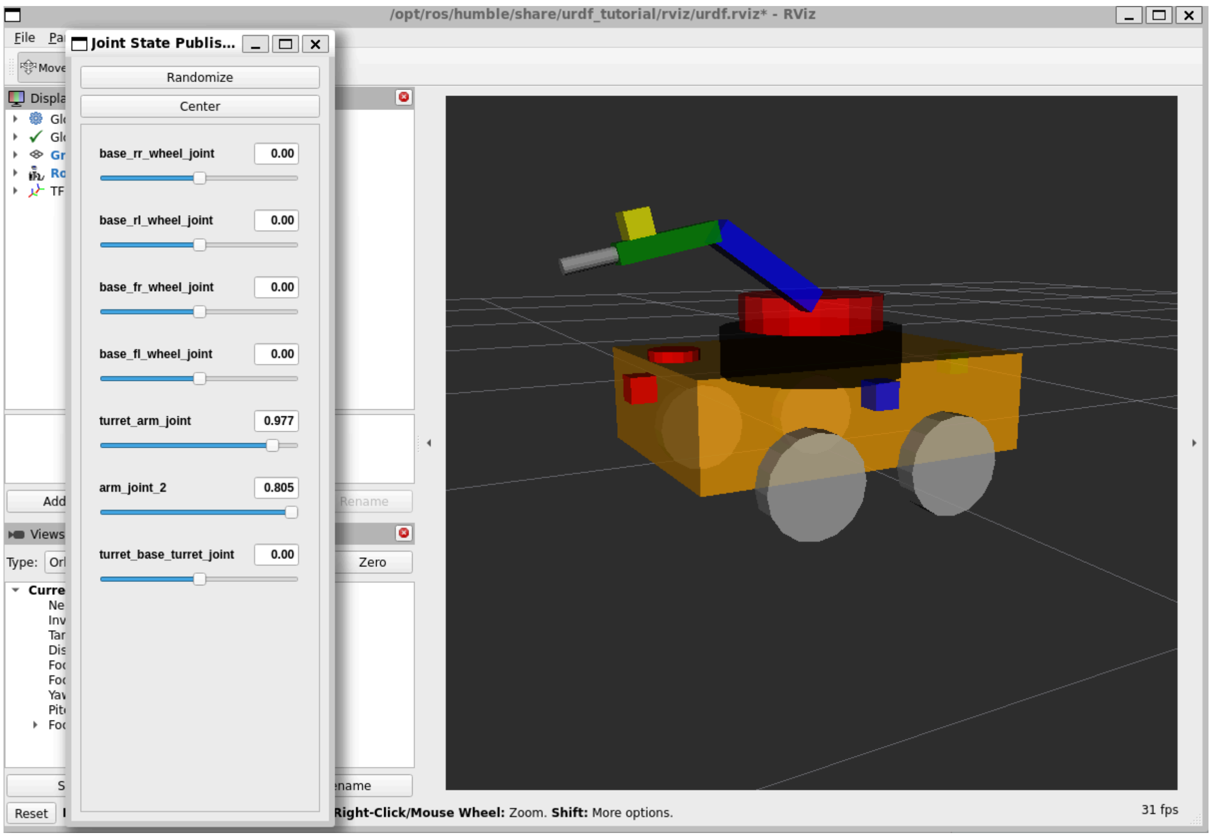
Fig 4.3.3 Sensor feedback and visualisations(2)

4.4 ROS 2 Integration and TF Management

- The **robot_state_publisher** and **joint_state_publisher** nodes function as expected, broadcasting joint states and TF transforms for all robot components.

- The complete **TF tree**, including frames for the base, wheels, turret, cameras, and LIDAR, is properly established and visualized in RViz.

- **Communication between ROS 2 nodes** and **Gazebo plugins** (for control and perception) is verified, ensuring seamless integration across the system.


/opt/ros/humble/share/urdf_tutorial/rviz/urdf.rviz* - RViz

Joint State Publis...

base_rr_wheel_joint	0.00
base_rl_wheel_joint	0.00
base_fr_wheel_joint	0.00
base_fl_wheel_joint	0.00
turret_arm_joint	0.977
arm_joint_2	0.805
turret_base_turret_joint	0.00

Right-Click/Mouse Wheel: Zoom. Shift: More options.

31 fps

Fig 4.4.1 ROS 2 Integration and TF Management

4.5 Challenges Faced

During the development of the **ROS 2–based Water Turret AGV**, several technical challenges were encountered and addressed:

- **TF Tree Configuration:**
  Ensuring correct parent/child relationships in the complex TF tree, especially with multiple cameras, was essential to prevent broken transforms and maintain consistent spatial relationships between frames.

- **Camera Plugin Parameters:**
  Tuning parameters such as field of view (FOV), near and far clipping planes, and update rates was critical for achieving accurate depth perception and ensuring reliable obstacle detection and fire identification.

- **Controller Gain Tuning:**
  Adjusting effort and velocity limits on continuous joints, including those of the arm and turret, was necessary to prevent oscillations and achieve smooth, stable control of articulated components.

- **Publishing to Arm and Turret Links:**
  Publishing appropriate joint states and command topics to the arm and turret links presented challenges in maintaining proper actuation of these subsystems within the simulation, especially under complex motion sequences.

- **ROS 2–Gazebo Compatibility:**
  Synchronizing plugin versions between ROS 2 Rolling and Gazebo was essential to ensure compatibility and stability of the simulation environment. Mismatched versions caused issues such as simulation crashes and plugin errors.

- **Integration Complexity:**
  Combining navigation, perception, and actuation within a single ROS 2 framework required careful synchronization of node lifecycles, message passing, and system resources to achieve reliable operation.

6.0 CONCLUSION

This project successfully demonstrates the design, simulation, and integration of a **ROS 2–based Water Turret Autonomous Ground Vehicle (AGV)**, capable of detecting and responding to fires in a simulated environment. The system combines several core components:

- **Multi-Camera Perception:** Leveraging multi-camera vision enables the detection of fires with high spatial awareness, providing robust situational perception.

- **Differential-Drive Navigation:** The AGV navigates through complex environments using a differential drive system, ensuring accurate positioning and mobility.

- **Articulated Water Turret Actuation:** An articulated turret is implemented for precise targeting and simulated water delivery, enabling an effective response to fire events.

- **ROS 2 Integration:** The use of ROS 2 facilitates modularity, real-time data exchange, and scalability, forming a solid foundation for future system upgrades and deployment.

- **XACRO-Based Robot Design:** The modular XACRO (XML Macros) approach enables flexible robot configuration, easing both simulation in Gazebo and potential hardware implementation.

- **Plugin-Based Simulation:** The Gazebo simulation, enhanced by ROS 2 plugins, ensures a realistic test environment for validating system functionality before real-world deployment.

This project not only showcases a **working prototype in simulation** but also paves the way for **future development of autonomous firefighting robots** capable of operating in hazardous and dynamic environments. The modular architecture and plugin-based simulation approach ensure easy adaptation for various real-world scenarios, including the integration of new sensors, advanced control algorithms, and real-robot deployment.

In summary, the Water Turret AGV project exemplifies the successful **integration of navigation, perception, and actuation** in a simulated firefighting context, setting a solid foundation for **advancing autonomous fire response technologies**.

7.0 FUTURE SCOPE

Building upon the successful implementation of the simulated **Water Turret AGV**, several areas of improvement and expansion are identified to enhance system performance, realism, and readiness for real-world deployment:

- **Autonomous Navigation:**
  Integrate **Simultaneous Localization and Mapping (SLAM)** and **autonomous path planning (Nav2)** to enable the AGV to navigate dynamically in complex environments, allowing full autonomy without manual control.

- **Fire Detection Algorithms:**
  Incorporate **thermal imaging sensors** or **RGB-based flame detection algorithms** to enable accurate and automatic fire detection, improving the system's responsiveness to real-world fire events.

- **Real-World Deployment:**
  Transition the simulated system to physical hardware by exporting the **XACRO robot model** and ROS 2 control nodes, enabling testing and validation in real environments with a physical robot platform.

- **Advanced Fluid Simulation:**
  Enhance the realism of the firefighting simulation by employing **Gazebo's particle system** to simulate realistic water spray effects from the turret nozzle, including fluid dynamics and interaction with the environment.

- **Automatic Fire Detection and Response:**
  Develop a fully automated pipeline where the AGV can autonomously detect fires, plan optimal paths, and activate the turret to suppress the fire, without human intervention.

- **Nozzle Spray Simulation and Control:**
  Simulate dynamic nozzle spray behavior and implement advanced **turret control algorithms** to achieve accurate targeting and efficient fire suppression in simulation and, eventually, in real-world scenarios.

- **Scalability and Modularity:**
  Further modularize the robot's software and hardware components to enable seamless integration of additional sensors, actuators, and control algorithms for different fire scenarios and environmental challenges.